

**MUNIX**  
**PROGRAMMER'S MANUAL**  
**VOLUME 1**  
**APPENDIX**

# INDEX

PROCEEDINGS OF THE

CONFERENCE

ALBANY

1964  
CONFERENCE  
ALBANY  
1964

Information in this document is subject to change without notice and does not represent a commitment on the part of Periphere Computer Systeme GmbH. The software described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of the agreement.

Copyright 1979, 1980 Regents of the University of California

Permission to copy these documents or any portion thereof as necessary for licensed use of the software is granted to licensees of this software, provided this copyright notice and statement of permission are included.

Copyright 1982, Siemens AG

Siemens software products are copyrighted by and shall remain property of Siemens AG.

Copyright 1982, Periphere Computer Systeme GmbH

PCS software products are copyrighted by and shall remain property of PCS GmbH.







**NAME**

as - assembler

**SYNOPSIS**

as file ...

**DESCRIPTION**

As assembles the named files, which must have the suffix '.s'. The output of the assembly is left on the corresponding files with suffix '.o'. Assembler listings are produced on the corresponding files with suffix '.lst'.

**FILES**

/lib/cpp	C preprocessor
/lib/asm	assembler
/usr/tmp/asm*	temporary files
/lib/symfile	used for initialization of /lib/asm
name.s	assembler source
name.o	assembled module
name.lst	assembler listing

**SEE ALSO**

Ulrike Weng-Beckmann, *Assembler 68000 Users Guide*  
Motorola, *Macro Assembler Reference Manual*

**DIAGNOSTICS**

Diagnostics are given in the assembler listing.





## NAME

bfs - big file scanner

## SYNOPSIS

bfs [ - ] name

## DESCRIPTION

*Bfs* is (almost) like *ed*(1) except that it is read-only and processes much larger files. Files can be up to 1024K bytes (the maximum possible size) and 32K lines, with up to 255 characters per line. *Bfs* is usually more efficient than *ed* for scanning a file, since the file is not copied to a buffer. It is most useful for identifying sections of a large file where *csplit*(1) can be used to divide it into more manageable pieces for editing.

Normally, the size of the file being scanned is printed, as is the size of any file written with the *w* command. The optional - suppresses printing of sizes. Input is prompted with • if P and a carriage return are typed as in *ed*. Prompting can be turned off again by inputting another P and carriage return. Note that messages are given in response to errors if prompting is turned on.

All address expressions described under *ed* are supported. In addition, regular expressions may be surrounded with two symbols besides / and ?: > indicates downward search without wrap-around, and < indicates upward search without wrap-around. Since *bfs* uses a different regular expression-matching routine from *ed*, the regular expressions accepted are slightly wider in scope (see *regex*(3X)). There is a slight difference in mark names: only the letters a through z may be used, and all 26 marks are remembered.

The *e*, *g*, *v*, *k*, *n*, *p*, *q*, *w*, *=*, *!* and null commands operate as described under *ed*. Commands such as *---*, *+++*, *+++*, *-12*, and *+4p* are accepted. Note that *1,10p* and *1,10* will both print the first ten lines. The *f* command only prints the name of the file being scanned; there is no *remembered* file name. The *w* command is independent of output diversion, truncation, or crunching (see the *xo*, *xt* and *xc* commands, below). The following additional commands are available:

***xf file***

Further commands are taken from the named *file*. When an end-of-file is reached, an interrupt signal is received or an error occurs, reading resumes with the file containing the *xf*. *Xf* commands may be nested to a depth of 10.

***xo [file]***

Further output from the *p* and null commands is diverted to the named *file*, which, if necessary, is created mode 666. If *file* is missing, output is diverted to the standard output. Note that each diversion causes truncation or creation of the file.

***:label***

This positions a *label* in a command file. The *label* is terminated by new-line, and blanks between the *:* and the start of the *label* are ignored. This command may also be used to insert comments into a command file, since labels need not be referenced.



The first part of the document is a letter from the President of the United States to the Congress, dated January 3, 1862. The letter is signed by Abraham Lincoln and is addressed to the Senate and House of Representatives. The letter discusses the state of the Union and the progress of the war against the Confederacy.

The second part of the document is a report from the Secretary of the War Department, dated January 10, 1862. The report is signed by Edwin M. Stanton and is addressed to the President. The report discusses the military situation and the progress of the war.

The third part of the document is a report from the Secretary of the Treasury, dated January 15, 1862. The report is signed by Alexander C. Harris and is addressed to the President. The report discusses the financial situation and the progress of the war.

The fourth part of the document is a report from the Secretary of the Navy, dated January 20, 1862. The report is signed by Gideon Welles and is addressed to the President. The report discusses the naval situation and the progress of the war.

The fifth part of the document is a report from the Secretary of the Interior, dated January 25, 1862. The report is signed by Caleb B. Smith and is addressed to the President. The report discusses the land situation and the progress of the war.

The sixth part of the document is a report from the Secretary of the War, dated February 1, 1862. The report is signed by Edwin M. Stanton and is addressed to the President. The report discusses the military situation and the progress of the war.

The seventh part of the document is a report from the Secretary of the Navy, dated February 5, 1862. The report is signed by Gideon Welles and is addressed to the President. The report discusses the naval situation and the progress of the war.

(...) **xb/regular expression/label**

A jump (either upward or downward) is made to *label* if the command succeeds. It fails under any of the following conditions:

1. Either address is not between 1 and 3.
2. The second address is less than the first.
3. The regular expression doesn't match at least one line in the specified range, including the first and last lines.

On success, . is set to the line matched and a jump is made to *label*. This command is the only one that doesn't issue an error message on bad addresses, so it may be used to test whether addresses are bad before other commands are executed. Note that the command

**xb/~ /label**

is an unconditional jump.

The **xb** command is allowed only if it is read from someplace other than a terminal. If it is read from a pipe only a downward jump is possible.

**xt number**

Output from the **p** and null commands is truncated to at most *number* characters. The initial number is 255.

**xv[digit][spaces][value]**

The variable name is the specified *digit* following the **xv**. **xv5100** or **xv5 100** both assign the value 100 to the variable 5. **Xv61,100p** assigns the value 1,100p to the variable 6. To reference a variable, put a % in front of the variable name. For example, using the above assignments for variables 5 and 6:

```
1,%5p
1,%5
%6
```

will all print the first 100 lines.

**g/%5/p**

would globally search for the characters 100 and print each line containing a match. To escape the special meaning of %, a \ must precede it.

**g/".\*\%[cds]/p**

could be used to match and list lines containing *printf* of characters, decimal integers, or strings.

Another feature of the **xv** command is that the first line of output from a UNIX command can be stored into a variable. The only requirement is that the first character of *value* be an !. For example:

```
xv5!cat junk
!rm junk
!echo "%5"
```



The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also outlines the specific requirements for record-keeping, including the need to maintain records for a minimum of five years and to ensure that records are easily accessible and retrievable.

The second part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also outlines the specific requirements for record-keeping, including the need to maintain records for a minimum of five years and to ensure that records are easily accessible and retrievable.

The third part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also outlines the specific requirements for record-keeping, including the need to maintain records for a minimum of five years and to ensure that records are easily accessible and retrievable.

The fourth part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The document also outlines the specific requirements for record-keeping, including the need to maintain records for a minimum of five years and to ensure that records are easily accessible and retrievable.



**xv6!expr %6 + 1**

would put the current line into variable 5, print it, and increment the variable 6 by one. To escape the special meaning of ! as the first character of *value*, precede it with a \.

**xv7\!date**

stores the value !date into variable 7.

**xbz label**

**xbn label**

These two commands will test the last saved *return code* from the execution of a UNIX command (!*command*) or nonzero value, respectively, to the specified label. The two examples below both search for the next five lines containing the string *size*.

```
xv55
:l
/size/
xv5!expr %5 - 1
!if 0%5 != 0 exit 2
xbn l
xv45
:l
/size/
xv4!expr %4 - 1
!if 0%4 = 0 exit 2
xbz l
```

**xc [switch]**

If *switch* is 1, output from the p and null commands is crunched; if *switch* is 0 it isn't. Without an argument, **xc** reverses *switch*. Initially *switch* is set for no crunching. Crunched output has strings of tabs and blanks reduced to one blank and blank lines suppressed.

SEE ALSO

csplit(1), ed(1), regex(3X).

## 10153

The first of these is the fact that the  
the first of these is the fact that the  
the first of these is the fact that the

## 10154

The first of these is the fact that the

## 10155

The first of these is the fact that the  
the first of these is the fact that the  
the first of these is the fact that the

## 10156

The first of these is the fact that the

## 10157

The first of these is the fact that the

## 10158

The first of these is the fact that the

## 10159

The first of these is the fact that the  
the first of these is the fact that the  
the first of these is the fact that the

## 10160

The first of these is the fact that the

## 10161

The first of these is the fact that the

## 10162

The first of these is the fact that the

## 10163

The first of these is the fact that the

## 10164

The first of these is the fact that the

## 10165

The first of these is the fact that the

## 10166

The first of these is the fact that the

## 10167

The first of these is the fact that the

## 10168

The first of these is the fact that the

**DIAGNOSTICS**

? for errors in commands, if prompting is turned off. Self-explanatory error messages when prompting is on.



1907

1907

1907

The following is a list of the names of the persons who have been elected to the office of the President of the United States since the year 1789.



## NAME

**bs** — a compiler/interpreter for modest-sized programs

## SYNOPSIS

**bs** [ *file* [ *args* ] ]

## DESCRIPTION

*Bs* is a remote descendant of Basic and Snobol4 with a little C language thrown in. *Bs* is designed for programming tasks where program development time is as important as the resulting speed of execution. Formalities of data declaration and file/process manipulation are minimized. Line-at-a-time debugging, the *trace* and *dump* statements, and useful run-time error messages all simplify program testing. Furthermore, incomplete programs can be debugged; *inner* functions can be tested before *outer* functions have been written and vice versa.

If the command line *file* argument is provided, the file is used for input before the console is read. By default, statements read from the file argument are compiled for later execution. Likewise, statements entered from the console are normally executed immediately (see *compile* and *execute* below). Unless the final operation is assignment, the result of an immediate expression statement is printed.

*Bs* programs are made up of input lines. If the last character on a line is a \, the line is continued. *Bs* accepts lines of the following form:

statement  
label statement

A label is a *name* (see below) followed by a colon. A label and a variable can have the same name.

A *bs* statement is either an expression or a keyword followed by zero or more expressions. Some keywords (*clear*, *compile*, *!*, *execute*, *include*, *ibase*, *obase*, and *run*) are always executed as they are compiled.

## Statement Syntax:

## expression

The expression is executed for its side effects (value, assignment or function call). The details of expressions follow the description of statement types below.

## break

*Break* exits from the inner-most *for/while* loop.

## clear

Clears the symbol table and compiled statements. *Clear* is executed immediately.

compile [ *expression* ]

Succeeding statements are compiled (overrides the immediate execution default). The optional *expression* is evaluated and used as a file name for further input. A *clear* is associated with this latter case. *Compile* is executed immediately.

## continue

*Continue* transfers to the loop-continuation of the current *for/while* loop.



1. The first part of the paper is devoted to the study of the

2. The second part of the paper is devoted to the study of the

3. The third part of the paper is devoted to the study of the

4. The fourth part of the paper is devoted to the study of the

5. The fifth part of the paper is devoted to the study of the

6. The sixth part of the paper is devoted to the study of the

7. The seventh part of the paper is devoted to the study of the

8. The eighth part of the paper is devoted to the study of the

9. The ninth part of the paper is devoted to the study of the

10. The tenth part of the paper is devoted to the study of the

11. The eleventh part of the paper is devoted to the study of the

12. The twelfth part of the paper is devoted to the study of the

13. The thirteenth part of the paper is devoted to the study of the

14. The fourteenth part of the paper is devoted to the study of the

15. The fifteenth part of the paper is devoted to the study of the

16. The sixteenth part of the paper is devoted to the study of the



**dump**

The name and current value of every non-local variable is printed. After an error or interrupt, the number of the last statement and (possibly) the user-function trace are displayed.

**exit [ expression ]**

Return to system level. The expression is returned as process status.

**execute**

Change to immediate execution mode (an interrupt has a similar effect). This statement does not cause stored statements to execute (see *run* below).

**for** name = expression expression statement

**for** name = expression expression

...

**next**

**for** expression , expression , expression statement

**for** expression , expression , expression

...

**next**

The *for* statement repetitively executes a statement (first form) or a group of statements (second form) under control of a named variable. The variable takes on the value of the first expression, then is incremented by one on each loop, not to exceed the value of the second expression. The third and fourth forms require three expressions separated by commas. The first of these is the initialization, the second is the test (true to continue), and the third is the loop-continuation action (normally an increment).

**fun** f([ a, ... ]) [ v, ... ]

...

**nuf**

*Fun* defines the function name, arguments, and local variables for a user-written function. Up to ten arguments and local variables are allowed. Such names cannot be arrays, nor can they be I/O associated. Function definitions may not be nested.

**freturn**

A way to signal the failure of a user-written function. See the interrogation operator (?) below. If interrogation is not present, *freturn* merely returns zero. When interrogation is active, *freturn* transfers to that expression (possibly by-passing intermediate function returns).

**ibase N**

*Ibase* sets the input base (radix) to *N*. The only supported values for *N* are 8, 10 (the default), and 16. Hexadecimal values 10-15 are entered as a-f. A leading digit is required (i.e., f0a must be entered as 0f0a). *Ibase* (and *obase*, below) are executed immediately.

**goto name**

Control is passed to the internally stored statement with the matching label.

THE UNIVERSITY OF CHICAGO  
DIVISION OF THE PHYSICAL SCIENCES  
DEPARTMENT OF CHEMISTRY

RECEIVED BY THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964

FROM THE LIBRARY OF THE UNIVERSITY OF CHICAGO  
JANUARY 10, 1964



**if** expression statement  
**if** expression

...  
 [ **else**  
 ... ]

**fi**

The statement (first form) or group of statements (second form) is executed if the expression evaluates to non-zero. The strings 0 and "" (null) evaluate as zero. In the second form, an optional *else* allows for a group of statements to be executed when the first group is not. The only statement permitted on the same line with an *else* is an *if*; only other *fi*'s can be on the same line with a *fi*. The elision of *else* and *if* into an *elif* is supported. Only a single *fi* is required to close an *if* ... *elif* ... [ *else* ... ] sequence.

**include** expression

The expression must evaluate to a file name. The file must contain *bs* statements. *include* statements may not be nested.

**obase** *N*

*Obase* sets the input base to *N* (see *ibase* above).

**onintr** label

**onintr**

The *onintr* command provides program control of interrupts. In the first form, control will pass to the label given, just as if a *goto* had been executed at the time *onintr* was executed. The effect of the statement is cleared after each interrupt. In the second form, an interrupt will cause *bs* to terminate.

**return** [expression]

The expression is evaluated and the result is passed back as the value of a function call. If no expression is given, zero is returned.

**run**

The random number generator is reset. Control is passed to the first internal statement. If the *run* statement is contained in a file, it should be the last statement.

**stop**

Execution of internal statements is stopped. *Bs* reverts to immediate mode.

**trace** [ expression ]

The *trace* statement controls function tracing. If the expression is null (or evaluates to zero), tracing is turned off. Otherwise, a record of user-function calls/returns will be printed. Each *return* decrements the *trace* expression value.

**while** expression statement

**while** expression

...  
**next**

*While* is similar to *for* except that only the conditional expression for loop-continuation is given.

1954-1955  
1956-1957

1958

1959

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1954-1955. The names are listed in alphabetical order of their last names.

1954-1955

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1955-1956. The names are listed in alphabetical order of their last names.

1955-1956

1956-1957

1957-1958

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1956-1957. The names are listed in alphabetical order of their last names.

1956-1957

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1957-1958. The names are listed in alphabetical order of their last names.

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1958-1959. The names are listed in alphabetical order of their last names.

1958-1959

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1959-1960. The names are listed in alphabetical order of their last names.

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1960-1961. The names are listed in alphabetical order of their last names.

1960-1961  
1961-1962

The following is a list of the names of the persons who have been elected to the office of President of the American Society of Zoologists for the year 1961-1962. The names are listed in alphabetical order of their last names.

1961-1962

1962-1963



**! shell command**

An immediate escape to the Shell.

**# ...**

This statement is ignored. It is used to interject commentary in a program.

**Expression Syntax:****name**

A name is used to specify a variable. Names are composed of a letter (upper or lower case) optionally followed by letters and digits. Only the first six characters of a name are significant. Except for names declared in *fun* statements, all names are global to the program. Names can take on numeric (double float) values, string values, or can be associated with input/output (see the built-in function *open()* below).

**name ( [expression [ , expression] ... ] )**

Functions can be called by a name followed by the arguments in parentheses separated by commas. Except for built-in functions (listed below), the name must be defined with a *fun* statement. Arguments to functions are passed by value.

**name [ expression [ , expression ] ... ]**

This syntax is used reference either arrays or tables (see built-in *table* functions below). For arrays, each expression is truncated to an integer and used as a specifier for the name. The resulting array reference is syntactically identical to a name; *a[1,2]* is the same as *a[1][2]*. The truncated expressions are restricted to values between 0 and 32767.

**number**

A number is used to represent a constant value. A number is written in Fortran style, and contains digits, an optional decimal point, and possibly a scale factor consisting of an *e* followed by a possibly signed exponent.

**string**

Character strings are delimited by " characters. The \ escape character allows the double quote (\"), new-line (\n), carriage return (\r), backspace (\b), and tab (\t) characters to appear in a string. Otherwise, \ stands for itself.

**( expression )**

Parentheses are used to alter the normal order of evaluation.

**( expression, expression [ , expression ... ] ) [ expression ]**

The bracketed expression is used as a subscript to select a comma-separated expression from the parenthesized list. List elements are numbered from the left, starting at zero. The expression:

( False, True ) [ a == b ]

has the value True if the comparison is true.

**? expression**

The interrogation operator tests for the success of the expression rather than its value. At the moment, it is useful for testing end-of-

100 - 1000

1000 - 10000

10000 - 100000

100000 - 1000000

1000000 - 10000000

10000000 - 100000000

100000000 - 1000000000

1000000000 - 10000000000

10000000000 - 100000000000

100000000000 - 1000000000000

1000000000000 - 10000000000000

10000000000000 - 100000000000000

100000000000000 - 1000000000000000

1000000000000000 - 10000000000000000

10000000000000000 - 100000000000000000

100000000000000000 - 1000000000000000000

1000000000000000000 - 10000000000000000000



file (see examples in the *Programming Tips* section below), the result of the *eval* built-in function, and for checking the return from user-written functions (see *freturn*). An interrogation "trap" (end-of-file, etc.) causes an immediate transfer to the most recent interrogation, possibly skipping assignment statements or intervening function levels.

- expression

The result is the negation of the expression.

++ name

Increments the value of the variable (or array reference). The result is the new value.

-- name

Decrements the value of the variable. The result is the new value.

! expression

The logical negation of the expression. Watch out for the shell escape command.

expression operator expression

Common functions of two arguments are abbreviated by the two arguments separated by an operator denoting the function. Except for the assignment, concatenation, and relational operators, both operands are converted to numeric form before the function is applied.

**Binary Operators** (in increasing precedence):

=

= is the assignment operator. The left operand must be a name or an array element. The result is the right operand. Assignment binds right to left, all other operators bind left to right.

\_

\_ (underscore) is the concatenation operator.

& |

& (logical and) has result zero if either of its arguments are zero. It has result one if both of its arguments are non-zero; | (logical or) has result zero if both of its arguments are zero. It has result one if either of its arguments is non-zero. Both operators treat a null string as a zero.

< <= > >= == !=

The relational operators (< less than, <= less than or equal, > greater than, >= greater than or equal, == equal to, != not equal to) return one if their arguments are in the specified relation. They return zero otherwise. Relational operators at the same level extend as follows: *a>b>c* is the same as *a>b & b>c*. A string comparison is made if both operands are strings.

+ -

Add and subtract.

\* / %

Multiply, divide, and remainder.





^ Exponentiation.

### Built-in Functions:

#### *Dealing with arguments*

##### **arg(i)**

is the value of the *i*-th actual parameter on the current level of function call. At level zero, *arg* returns the *i*-th command-line argument (*arg*(0) returns *bs*).

##### **narg()**

returns the number of arguments passed. At level zero, the command argument count is returned.

#### *Mathematical*

##### **abs(x)**

is the absolute value of *x*.

##### **atan(x)**

is the arctangent of *x*. Its value is between  $-\pi/2$  and  $\pi/2$ .

##### **ceil(x)**

returns the smallest integer not less than *x*.

##### **cos(x)**

is the cosine of *x* (radians).

##### **exp(x)**

is the exponential function of *x*.

##### **floor(x)**

returns the largest integer not greater than *x*.

##### **log(x)**

is the natural logarithm of *x*.

##### **rand()**

is a uniformly distributed random number between zero and one.

##### **sin(x)**

is the sine of *x* (radians).

##### **sqrt(x)**

is the square root of *x*.

#### *String operations*

##### **size(s)**

the size (length in bytes) of *s* is returned.

##### **format(f, a)**

returns the formatted value of *a*. *F* is assumed to be a format specification in the style of *printf*(3S). Only the *%...f*, *%...e*, and *%...s* types are safe.

##### **index(x, y)**

returns the number of the first position in *x* that any of the characters from *y* matches. No match yields zero.

##### **trans(s, f, t)**

Translates characters of the source *s* from matching characters in *f*





to a character in the same position in *t*. Source characters that do not appear in *f* are copied to the result. If the string *f* is longer than *t*, source characters that match in the excess portion of *f* do not appear in the result.

**substr(*s*, *start*, *width*)**

returns the sub-string of *s* defined by the *starting* position and *width*.

**match(*string*, *pattern*)**

**mstring(*n*)**

The *pattern* is similar to the regular expression syntax of the *ed*(1) command. The characters *.*, *[*, *]*, *^* (inside brackets), *\** and *\$* are special. The *mstring* function returns the *n*-th ( $1 \leq n \leq 10$ ) sub-string of the subject that occurred between pairs of the pattern symbols *\(* and *\)* for the most recent call to *match*. To succeed, patterns must match the beginning of the string (as if all patterns began with *^*). The function returns the number of characters matched. For example:

```
match("a123ab123", ".*\([a-z]\)") == 6
mstring(1) == "b"
```

### File handling

**open(*name*, *file*, *function*)**

**close(*name*)**

The *name* argument must be a *bs* variable name (passed as a string). For the *open*, the *file* argument may be 1) a 0 (zero), 1, or 2 representing standard input, output, or error output, respectively, 2) a string representing a file name, or 3) a string beginning with an *!* representing a command to be executed (via *sh -c*). The *function* argument must be either *r* (read), *w* (write), *W* (write without new-line), or *a* (append). After a *close*, the *name* reverts to being an ordinary variable. The initial associations are:

```
open("get", 0, "r")
open("put", 1, "w")
open("puterr", 2, "w")
```

Examples are given in the following section.

**access(*s*, *m*)**

executes *access*(2).

**ftype(*s*)**

returns a single character file type indication: *f* for regular file, *d* for directory, *b* for block special, or *c* for character special.

### Tables

**table(*name*, *size*)**

A table in *bs* is an associatively accessed, single-dimension array. "Subscripts" (called keys) are strings (numbers are converted). The *name* argument must be a *bs* variable name (passed as a string). The *size* argument sets the minimum number of elements to be allocated. *Bs* prints an error message and stops on table overflow.

**item(*name*, *i*)**

1. The first section of the report deals with the general situation of the country and the progress of the work during the year.

2. The second section deals with the results of the work done during the year, and the progress of the various projects.

3. The third section deals with the financial statement of the work, and the results of the various projects.

4. The fourth section deals with the results of the work done during the year, and the progress of the various projects.

5. The fifth section deals with the financial statement of the work, and the results of the various projects.

6. The sixth section deals with the results of the work done during the year, and the progress of the various projects.

7. The seventh section deals with the financial statement of the work, and the results of the various projects.

8. The eighth section deals with the results of the work done during the year, and the progress of the various projects.

9. The ninth section deals with the financial statement of the work, and the results of the various projects.



**key()**

The *item* function accesses table elements sequentially (in normal use, there is no orderly progression of key values). Where the *item* function accesses values, the *key* function accesses the "subscript" of the previous *item* call. The *name* argument should not be quoted. Since exact table sizes are not defined, the interrogation operator should be used to detect end-of-table, for example:

```
table("t", 100)
```

```
...
# If word contains "party", the following expression adds one to
the count
# of that word:
++t[word]
```

```
...
# To print out the the key/value pairs:
for i = 0, ?(s = item(t, i)), ++i if key() put = key()_"":_s
```

**iskey(name, word)**

The *iskey* function tests whether the key word exists in the table *name* and returns one for true, zero for false.

*Odds and ends***eval(s)**

The string argument is evaluated as a *bs* expression. The function is handy for converting numeric strings to numeric internal form. *Eval* can also be used as a crude form of indirection, as in:

```
name = "xyz"
eval("++" _ name)
```

which increments the variable *xyz*. In addition, *eval* preceded by the interrogation operator permits the user to control *bs* error conditions. For example:

```
?eval("open(\"X\", \"XXX\", \"r\")")
```

returns the value zero if there is no file named "XXX" (instead of halting the user's program). The following executes a *goto* to the label *L* (if it exists):

```
label="L"
if !(?eval("goto " _ label)) puterr = "no label"
```

**plot(request, args)**

The *plot* function produces output on devices recognized by *tplot(1G)*. The *requests* are as follows:

<i>Call</i>	<i>Function</i>
plot(0, term)	causes further <i>plot</i> output to be piped into <i>tplot(1G)</i> with an argument of <i>-Tterm</i> .
plot(1)	"erases" the plotter.
plot(2, string)	labels the current point with <i>string</i> .
plot(3, x1, y1, x2, y2)	draws the line between (x1,y1) and (x2,y2).

1941  
The first of the year was a very busy one for the  
company. The new product was introduced and  
the sales were very good. The company was  
very successful in the first quarter of the year.

The second quarter was also very successful. The  
company was able to increase its sales and  
the new product was well received by the  
customers.

The third quarter was a very busy one for the  
company. The new product was introduced and  
the sales were very good. The company was  
very successful in the third quarter of the year.

The fourth quarter was also very successful. The  
company was able to increase its sales and  
the new product was well received by the  
customers.

The fifth quarter was a very busy one for the  
company. The new product was introduced and  
the sales were very good. The company was  
very successful in the fifth quarter of the year.

The sixth quarter was also very successful. The  
company was able to increase its sales and  
the new product was well received by the  
customers.

The seventh quarter was a very busy one for the  
company. The new product was introduced and  
the sales were very good. The company was  
very successful in the seventh quarter of the year.

The eighth quarter was also very successful. The  
company was able to increase its sales and  
the new product was well received by the  
customers.



plot(4, x, y, r)	draws a circle with center (x,y) and radius r.
plot(5, x1, y1, x2, y2, x3, y3)	draws an arc (counterclockwise) with center (x1,y1) and endpoints (x2,y2) and (x3,y3).
plot(6)	is not implemented.
plot(7, x, y)	makes the current point (x,y).
plot(8, x, y)	draws a line from the current point to (x,y).
plot(9, x, y)	draws a point at (x,y).
plot(10, string)	sets the line mode to <i>string</i> .
plot(11, x1, y1, x2, y2)	makes (x1,y1) the lower left corner of the plotting area and (x2,y2) the upper right corner of the plotting area.
plot(12, x1, y1, x2, y2)	causes subsequent x (y) coordinates to be multiplied by x1 (y1) and then added to x2 (y2) before they are plotted. The initial scaling is plot(12, 1.0, 1.0, 0.0, 0.0).

Some requests do not apply to all plotters. All requests except zero and twelve are implemented by piping characters to *tplot(1G)*. See *plot(5)* for more details.

#### last()

in immediate mode, *last* returns the most recently computed value.

#### PROGRAMMING TIPS

Using *bs* as a calculator:

```
$ bs
# Distance (inches) light travels in a nanosecond.
186000 * 5280 * 12 / 1e9
11.78496
...

# Compound interest (6% for 5 years on $1,000).
int = .06 / 4
bal = 1000
for i = 1 5*4 bal = bal + bal*int
bal - 1000
346.855007
...
exit
```

The outline of a typical *bs* program:

```
# initialize things:
var1 = 1
open("read", "infile", "r")
...
```





```
# compute:
while ?(str = read)
    ...
next
# clean up:
close("read")
...
# last statement executed (exit or stop):
exit
# last input line:
run
```

Input/Output examples:

```
# Copy "oldfile" to "newfile".
open("read", "oldfile", "r")
open("write", "newfile", "w")
...
while ?(write = read)
    ...
# close "read" and "write":
close("read")
close("write")

# Pipe between commands.
open("ls", "!ls *", "r")
open("pr", "!pr -2 -h 'List'", "w")
while ?(pr = ls) ...
...
# be sure to close (wait for) these:
close("ls")
close("pr")
```

SEE ALSO

ed(1), sh(1), tplot(1G), access(2), printf(3S), stdio(3S), Section 3 of this volume for further description of the mathematical functions (pow(3M) is used for exponentiation), plot(5). *Bs* uses the Standard Input/Output package.





## NAME

cc, lcc - C compiler

## SYNOPSIS

cc [ option ] ... file ...

## DESCRIPTION

Cc is the UNIX C compiler. It accepts several types of arguments:

Arguments whose names end with '.c' are taken to be C source programs; they are compiled, and each object program is left on the file whose name is that of the source with '.o' substituted for '.c'. The '.o' file is normally deleted, however, if a single C program is compiled and linked all at one go. This C-compiler has the identifier "m68000" predefined, such that "#ifdef m68000" is true.

A call "cc test.c" is the same as "cc -c test.c && ld /lib/crt0.o test.o -lc && rm test.o". The call "cc \*.o" is the same as "ld /lib/crt0.o \*.o -lc". If your directory contains the file xyz.c, a call "make xyz" will result in "cc -o xyz xyz.c".

The following options are interpreted by cc. See ld(1) for link-time options.

- c Suppress the linking phase of the compilation, and force an object file to be produced even if only one program is compiled.
- o *output* Name the final output file *output*. If this option is used the file 'a.out' will be left undisturbed. This option is passed on to the linker.
- f This option must be given, if your program includes floating point operations. The library libffp.a will be searched before libc.a.
- S Compile the named C programs, and produce an assembly listing on corresponding files suffixed '.s'. This option is used mainly for compiler debugging.
- a This option, followed by a number, e.g. 1234 (decimal), 01234 (octal) or 0x1234 (hex), together with option -S starts the program counter in the produced assembly-listing with the given number.
- L Arrange for the compiler to produce code which puts the current linenum into the stackframe during execution. The C-stacktrace of adb(1) (command \$c or \$C) will then give you for each active procedure the current linenum.
- g Adds entries to the symbol table of the produced .o module which indicate line numbers and corresponding code location. Used by the pbug source oriented debugger.
- p Arrange for the compiler to produce code which counts the number of times each routine is called; also, if linking takes place, replace the standard startup routine by one which automatically calls monitor(3) at the start and arranges to write out a mon.out file at normal termination of execution of the object program. An execution profile can then be generated by use of prof(1).





- T With this option initialized data is put into the text- rather than the data segment. Thus for programs like the shell, whose text segments are shared between many users, memory space can be saved. Apply this option only to modules with nothing but constant data declarations.
- u All characters will be treated as "unsigned char".
- 4 Changes the integer size from 2 (default) to 4. This makes programs which neglect the differences between int and pointer or int and long much more portable. On the other hand, the produced code is less efficient. When the name lcc is linked to cc, lcc is equivalent to cc -4. This option is also passed to the linker.
- 2 Like option -4. However, short parameters remain short and char parameters are only converted to short, not int, before they are pushed on the stack. This option is used for example to compile the system calls in the 4-byte-integer standard library /lib/libLc.a, which must interface to a 2-byte-integer world.
- 8 Currently, only the Motorola Fast Floating Point Package is available for floating point simulation. This package does not implement 64-bit floating point. Therefore, double is considered to be the same as float by default. Option -8 changes the default such that sizeof(double) = 8. The code generator will then produce calls to procedures dadd, dmul etc. for double addition and multiplication, which are not yet implemented.
- P Run only the macro preprocessor and place the result for each '.c' file in a corresponding '.i' file that has no '#' lines in it.
- E Run only the macro preprocessor and send the result to the standard output. The output is intended for compiler debugging; it is unacceptable as input to cc.
- Dname=def
- Dname  
Define the *name* to the preprocessor, as if by '#define'. If no definition is given, the name is defined as 1.
- Uname  
Remove any initial definition of *name*.
- Idir '#include' files whose names do not begin with '/' are always sought first in the directory of the *file* argument, then in directories named in -I options, then in directories on a standard list.
- Bstring  
Find substitute compiler passes in the files named *string* with the suffixes cpp, c0, c1 and c2. If *string* is empty, use a standard backup version.
- t[p012]  
Find only the designated compiler passes in the files whose names are constructed by a -B option. In the absence of a -B option, the *string* is taken to be '/usr/src/cmd/c/'.

Other arguments are taken to be either linker option arguments, or C-compatible object programs, typically produced by an earlier cc run, or

The first part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The second part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The third part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The fourth part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The fifth part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The sixth part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The seventh part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The eighth part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.

The ninth part of the paper is devoted to a discussion of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, which are based on the principle of the conservation of energy and the principle of the conservation of momentum.



perhaps libraries of C-compatible routines. These programs, together with the results of any compilations specified, are linked (in the order given) to produce an executable program with name **a.out**.

**FILES**

<b>file.c</b>	input file
<b>file.o</b>	object file
<b>a.out</b>	linked output
<b>/tmp/ctm?</b>	temporaries for cc
<b>/lib/cpp</b>	preprocessor
<b>/lib/c[012]</b>	compiler passes for cc
<b>/usr/c/oc[012]</b>	backup compiler passes for cc
<b>/usr/c/ocpp</b>	backup preprocessor
<b>/lib/crt0.o</b>	runtime initialization
<b>/lib/mcrt0.o</b>	runtime initialization for profiling
<b>/lib/libc.a</b>	standard library, see <i>intro</i> (3)
<b>/lib/libffp.a</b>	library with floating point routines
<b>/usr/include</b>	standard directory for '#include' files

**SEE ALSO**

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978  
D. M. Ritchie, *C Reference Manual*  
*monitor*(3), *prof*(1), *adb*(1), *ld*(1)

**DIAGNOSTICS**

The diagnostics produced by C itself are intended to be self-explanatory. Occasional messages may be produced by the linker.

**BUGS**

Option **-g** should also be usable by *adb*, so that breakpoints could be set to the start of a line.

1920

THE UNIVERSITY OF CHICAGO

1920

THE UNIVERSITY OF CHICAGO  
DEPARTMENT OF CHEMISTRY  
CHICAGO, ILL.

TO THE HONORABLE  
THE PRESIDENT OF THE UNIVERSITY OF CHICAGO  
FROM  
THE DEPARTMENT OF CHEMISTRY  
CHICAGO, ILL.

THE UNIVERSITY OF CHICAGO  
DEPARTMENT OF CHEMISTRY  
CHICAGO, ILL.

CHICAGO, ILL.

THE UNIVERSITY OF CHICAGO  
DEPARTMENT OF CHEMISTRY  
CHICAGO, ILL.

THE UNIVERSITY OF CHICAGO  
DEPARTMENT OF CHEMISTRY  
CHICAGO, ILL.



## NAME

**cpio** - copy file archives in and out

## SYNOPSIS

**cpio -o** [ **acBvS** ]

**cpio -i** [ **Bcdmrtuvs6S** ] [ **patterns** ]

**cpio -p** [ **adlmruv** ] **directory**

## DESCRIPTION

**Cpio -o** (copy out) reads the standard input to obtain a list of path names and copies those files onto the standard output together with path name and status information.

**Cpio -i** (copy in) extracts from the standard input (which is assumed to be the product of a previous **cpio -o**) the names of files selected by zero or more *patterns* given in the name-generating notation of *sh*(1). In *patterns*, meta-characters **?**, **\***, and **[...]** match the slash **/** character. The default for *patterns* is **\*** (i.e., select all files).

**Cpio -p** (pass) copies out and in in a single operation. Destination path names are interpreted relative to the named *directory*.

The meanings of the available options are:

- a** Reset access times of input files after they have been copied.
- B** Input/output is to be blocked 5,120 bytes to the record (does not apply to the *pass* option; meaningful only with data directed to or from **/dev/rmt?**).
- d** *Directories* are to be created as needed.
- c** Write *header* information in ASCII character form for portability.
- r** Interactively *rename* files. If the user types a null line, the file is skipped.
- t** Print a *table of contents* of the input. No files are created.
- u** Copy *unconditionally* (normally, an older file will not replace a newer file with the same name).
- v** *Verbose*: causes a list of file names to be printed. When used with the **t** option, the table of contents looks like the output of an **ls -l** command (see *ls*(1)).
- l** Whenever possible, link files rather than copying them. Usable only with the **-p** option.
- m** Retain previous file modification time. This option is ineffective on directories that are being copied.
- s** For transfer of a tape in *cpio* format between a PDP 11 and the QU68000. (The bytes are swapped except of the header information.)
- 6** Process an old (i.e., UNIX *Sixth* Edition format) file. Only useful with **-i** (copy in).
- S** set the blocking factor to 120 (speeds up I/O from resp. to the *streamer*).

## EXAMPLES

The first example below copies the contents of a directory into an archive; the second duplicates a directory hierarchy:

```
ls | cpio -o >/dev/mt0
```

The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that every entry, no matter how small, should be carefully documented to ensure the integrity of the financial data. This includes recording dates, amounts, and the nature of the transactions.

Secondly, the document highlights the need for regular reconciliation. By comparing internal records with external statements, discrepancies can be identified and corrected promptly. This process helps in maintaining the accuracy of the accounts and prevents errors from accumulating over time.

Thirdly, the document stresses the importance of transparency and accountability. All transactions should be clearly labeled and supported by appropriate documentation. This not only helps in tracking the flow of funds but also provides a clear audit trail for future reference.

Finally, the document concludes by stating that consistent adherence to these principles is essential for the long-term success and stability of any organization. It encourages a proactive approach to financial management, where potential issues are identified and addressed before they become significant problems.



```
cd olddir  
find . -print | cpio -pdl newdir
```

The trivial case "find . -print | cpio -oB >/dev/rmt0" can be handled more efficiently by:

```
find . -cpio /dev/rmt0
```

**SEE ALSO**

ar(1), find(1), cpio(5).

**BUGS**

Path names are restricted to 128 characters. If there are too many unique linked files, the program runs out of memory to keep track of them and, thereafter, linking information is lost. Only the super-user can copy special files.

1. The first part of the report is a summary of the work done during the year. It is a brief overview of the main results and conclusions of the research.

2. The second part of the report is a detailed account of the work done during the year. It is a more in-depth look at the research, including the methods used, the results obtained, and the conclusions drawn.



CREATE(1)

MUNIX

CREATE(1)

**NAME**

create - create a file

**SYNOPSIS**

create *file...*

**DESCRIPTION**

*Create* creates one or more files named *file...* with a length of 0 bytes.

10/10/10

10/10/10

10/10/10

10/10/10

10/10/10

10/10/10

10/10/10





**NAME**

**cref** - make cross-reference listing

**SYNOPSIS**

**cref** [ **-acilnostux123** ] files

**DESCRIPTION**

*Cref* makes a cross-reference listing of assembler or C programs; *files* are searched for symbols in the appropriate syntax.

The output report is in four columns:

1. symbol;
2. file name;
3. see below;
4. text as it appears in the file.

*Cref* uses either an *ignore* file or an *only* file. If the **-i** option is given, the next argument is taken to be an *ignore* file; if the **-o** option is given, the next argument is taken to be an *only* file. *Ignore* and *only* files are lists of symbols separated by new-lines. All symbols in an *ignore* file are ignored in columns 1 and 3 of the output. If an *only* file is given, only symbols in that file will appear in column 1. Only one of these options may be given; the default setting is **-i** using the default ignore file (see *FILES* below). Assembler pre-defined symbols or C keywords are ignored.

The **-s** option causes current symbols to be put in column 3. In the assembler, the current symbol is the most recent name symbol; in C, the current function name. The **-l** option causes the line number within the file to be put in column 3.

The **-t** option causes the next available argument to be used as the name of the intermediate file (instead of the temporary file */tmp/crt??*). This file is created and is *not* removed at the end of the process.

The *cref* options are:

- a** assembler format (default)
- c** C format input
- i** use an *ignore* file (see above)
- l** put line number in column 3 (instead of current symbol)
- n** omit column 4 (no context)
- o** use an *only* file (see above)
- s** current symbol in column 3 (default)
- t** user-supplied temporary file
- u** print only symbols that occur exactly once
- x** print only C external symbols
- 1** sort output on column 1 (default)
- 2** sort output on column 2
- 3** sort output on column 3.

**FILES**

*/tmp/crt??* temporaries  
*/usr/lib/cref/aign*  
default assembler *ignore* file  
*/usr/lib/cref/atab*  
grammar table for assembler files

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100



/usr/lib/cref/cign  
    default C *ignore* file  
/usr/lib/cref/ctab  
    grammar table for C files  
/usr/lib/cref/crpost  
    post-processor  
/usr/lib/cref/upost  
    post-processor for -u option

## SEE ALSO

as(1), cc(1), sort(1), xref(1).

## BUGS

*Cref* inserts an ASCII DEL character into the intermediate file after the eighth character of each name that is eight or more characters long in the source file.





## NAME

*csplit* - context split

## SYNOPSIS

*csplit* [-s] [-k] [-f prefix] file arg1 [... argn]

## DESCRIPTION

*Csplit* reads *file* and separates it into  $n+1$  sections, defined by the arguments *arg1* ... *argn*. By default the sections are placed in *xx00* ... *xxn* ( $n$  may not be greater than 99). These sections get the following pieces of *file*:

- 00: From the start of *file* up to (but not including) the line referenced by *arg1*.
- 01: From the line referenced by *arg1* up to the line referenced by *arg2*.
- ...
- $n+1$ : From the line referenced by *argn* to the end of *file*.

The options to *csplit* are:

- s *Csplit* normally prints the character counts for each file created. If the -s option is present, *csplit* suppresses the printing of all character counts.
- k *Csplit* normally removes created files if an error occurs. If the -k option is present, *csplit* leaves previously created files intact.
- f *prefix* If the -f option is used, the created files are named *prefix00* ... *prefixn*. The default is *xx00* ... *xxn*.

The arguments (*arg1* ... *argn*) to *csplit* can be a combination of the following:

*/rexp/*

A file is to be created for the section from the current line up to (but not including) the line containing the regular expression *rexp*. The current line becomes the line containing *rexp*. This argument may be followed by an optional + or - some number of lines (e.g., */Page/-5*).

%*rexp*%

This argument is the same as */rexp/*, except that no file is created for the section.

*lnno*

A file is to be created from the current line up to (but not including) *lnno*. The current line becomes *lnno*.

{*num*}

Repeat argument. This argument may follow any of the above arguments. If it follows a *rexp* type argument, that argument is applied *num* more times. If it follows *lnno*, the file will be split every *lnno* lines (*num* times) from that point.

Enclose all *rexp* type arguments that contain blanks or other characters meaningful to the Shell in the appropriate quotes. Regular expressions may not contain embedded new-lines. *Csplit* does not affect the original file; it is the users responsibility to remove it.





## EXAMPLES

```
csplit -f cobol file '/procedure division/' /par5./ /par16./
```

This example creates four files, cobol00 ... cobol03. After editing the "split" files, they can be recombined as follows:

```
cat cobol0[0-3] > file
```

Note that this example overwrites the original file.

```
csplit -k file 100 {99}
```

This example would split the file at every 100 lines, up to 10,000 lines. The -k option causes the created files to be retained if there are less than 10,000 lines; however, an error message would still be printed.

```
csplit -k prog.c '%main(%' '/~}/+1' {20}
```

Assuming that prog.c follows the normal C coding convention of ending routines with a } at the beginning of the line, this example will create a file containing each separate C routine (up to 21) in prog.c.

## SEE ALSO

ed(1), sh(1), regexp(7).

## DIAGNOSTICS

Self explanatory except for:

arg - out of range

which means that the given argument did not reference a line between the current position and the end of the file.





**NAME**

devnm - device name

**SYNOPSIS**

/etc/devnm [ names ]

**DESCRIPTION**

*Devnm* identifies the special file associated with the mounted file system where the argument *name* resides.

This command is most commonly used by */etc/rc* (see *rc(8)*) to construct a mount table entry for the root device.

**EXAMPLE**

The command:

    /etc/devnm /usr .

produces

    rp1 /usr

if */usr* is mounted on */dev/rp1*.

**FILES**

    /dev/rp\*

    /etc/mtab

**SEE ALSO**

    setmnt(1M).





## NAME

dump - incremental file system dump

## SYNOPSIS

dump [ key [ argument ... ] filesystem ]

## DESCRIPTION

*Dump* copies to magnetic tape all files changed after a certain date in the *filesystem*. The *key* specifies the date and other options about the dump. *Key* consists of characters from the set 0123456789fubsd.

- f** Place the dump on the next *argument* file instead of the tape.
- u** If the dump completes successfully, write the date of the beginning of the dump on file '/etc/ddate'. This file records a separate date for each filesystem and each dump level.
- 0-9** This number is the 'dump level'. All files modified since the last date stored in the file '/etc/ddate' for the same filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed; thus the option 0 causes the entire filesystem to be dumped.
- b** The size of the dump device is specified in number of blocks. The number is taken from the next *argument*. This option is useful when you dump to floppy disks. When the specified number of blocks is reached, the dump will wait for additional floppies.
- s** The size of the dump tape is specified in feet. The number of feet is taken from the next *argument*. When the specified size is reached, the dump will wait for reels to be changed. The default size is 2300 feet.
- d** The density of the tape, expressed in BPI, is taken from the next *argument*. This is used in calculating the amount of tape used per write. The default is 1600.

If no arguments are given, the *key* is assumed to be 9u and a default file system is dumped to the default tape.

Now a short suggestion on how perform dumps. Start with a full level 0 dump

dump 0u

Next, periodic level 9 dumps should be made on an exponential progression of tapes. (Sometimes called Tower of Hanoi - 1 2 1 3 1 2 1 4 ... tape 1 used every other time, tape 2 used every fourth, tape 3 used every eighth, etc.)

dump 9u

When the level 9 incremental approaches a full tape (about 78000 blocks at 1600 BPI blocked 20), a level 1 dump should be made.

dump 1u

After this, the exponential series should progress as uninterrupted. These level 9 dumps are based on the level 1 dump which is based on the level 0 full dump. This progression of levels of dump can be carried as far as desired.

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000



**FILES**

default filesystem and tape vary with installation.  
/etc/ddate: record dump dates of filesystem/level.

**SEE ALSO**

restor(1), dump(5), dumpdir(1)

**DIAGNOSTICS**

If the dump requires more than one tape, it will ask you to change tapes.  
Reply with a new-line when this has been done.

**BUGS**

Sizes are based on 1600 BPI blocked tape. The raw magtape device has to be used to approach these densities. Read errors on the filesystem are ignored. Write errors on the magtape are usually fatal.

RECEIVED BY THE DIRECTOR OF THE BUREAU OF THE CENSUS  
FROM THE CHIEF OF BUREAU OF THE CENSUS

100-100000

100-100000

100-100000

100-100000





## NAME

ed - text editor

## SYNOPSIS

ed [ - ] [ -x ] [ file ]

## DESCRIPTION

*Ed* is the standard text editor. If the *file* argument is given, *ed* simulates an *e* command (see below) on the named file; that is to say, the file is read into *ed*'s buffer so that it can be edited. The optional *-* suppresses the printing of character counts by *e*, *r*, and *w* commands, of diagnostics from *e* and *q* commands, and of the *!* prompt after a *!shell command*. If *-x* is present, an *x* command is simulated first to handle an encrypted file. *Ed* operates on a copy of the file it is editing; changes made to the copy have no effect on the file until a *w* (write) command is given. The copy of the text being edited resides in a temporary file called the *buffer*. There is only one buffer.

Commands to *ed* have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command*, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that the addresses can very often be omitted.

In general, only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While *ed* is accepting text, it is said to be in *input mode*. In this mode, *no* commands are recognized; all input is merely collected. Input mode is left by typing a period (.) alone at the beginning of a line.

*Ed* supports a limited form of *regular expression* notation; regular expressions are used in addresses to specify lines and in some commands (e.g., *s*) to specify portions of a line that are to be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE. The REs allowed by *ed* are constructed as follows:

The following *one-character REs* match a *single* character:

- 1.1 An ordinary character (*not* one of those discussed in 1.2 below) is a one-character RE that matches itself.
- 1.2 A backslash (\) followed by any special character is a one-character RE that matches the special character itself. The special characters are:
  - a. *., \*, [, and \* (period, asterisk, left square bracket, and backslash, respectively), which are always special, *except* when they appear within square brackets ([ ]); see 1.4 below).
  - b. *^* (caret or circumflex), which is special at the *beginning* of an *entire* RE (see 3.1 and 3.2 below), or when it immediately follows the left of a pair of square brackets ([ ]) (see 1.4 below).
  - c. *\$* (currency symbol), which is special at the *end* of an *entire* RE (see 3.2 below).
  - d. The character used to bound (i.e., delimit) an *entire* RE, which is special for that RE (for example, see how slash (/) is used in





the *g* command, below.)

- 1.3 A period (.) is a one-character RE that matches any character except new-line.
- 1.4 A non-empty string of characters enclosed in square brackets ([ ]) is a one-character RE that matches *any one* character in that string. If, however, the first character of the string is a circumflex (^), the one-character RE matches any character *except* new-line and the remaining characters in the string. The ^ has this special meaning *only* if it occurs first in the string. The minus (-) may be used to indicate a range of consecutive ASCII characters; for example, [0-9] is equivalent to [0123456789]. The - loses this special meaning if it occurs first (after an initial ^, if any) or last in the string. The right square bracket (]) does not terminate such a string when it is the first character within it (after an initial ^, if any); e.g., [ ]a-f] matches either a right square bracket (]) or one of the letters a through f inclusive. The four characters listed in 1.2.a above stand for themselves within such a string of characters.

The following rules may be used to construct REs from one-character REs:

- 2.1 A one-character RE is a RE that matches whatever the one-character RE matches.
- 2.2 A one-character RE followed by an asterisk (\*) is a RE that matches *zero* or more occurrences of the one-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.
- 2.3 A one-character RE followed by \{m\}, \{m,\}, or \{m,n\} is a RE that matches a *range* of occurrences of the one-character RE. The values of *m* and *n* must be non-negative integers less than 256; \{m\} matches *exactly* *m* occurrences; \{m,\} matches *at least* *m* occurrences; \{m,n\} matches *any number* of occurrences *between* *m* and *n* inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.
- 2.4 The concatenation of REs is a RE that matches the concatenation of the strings matched by each component of the RE.
- 2.5 A RE enclosed between the character sequences \ ( and \ ) is a RE that matches whatever the unadorned RE matches.
- 2.6 The expression \n matches the same string of characters as was matched by an expression enclosed between \ ( and \ ) *earlier* in the same RE. Here *n* is a digit; the sub-expression specified is that beginning with the *n*-th occurrence of \ ( counting from the left. For example, the expression ^(\.\*)\1\$ matches a line consisting of two repeated appearances of the same string.

Finally, an *entire* RE may be constrained to match only an initial segment or final segment of a line (or both):





- 3.1 A circumflex (^) at the beginning of an entire RE constrains that RE to match an *initial* segment of a line.
- 3.2 A currency symbol (\$) at the end of an entire RE constrains that RE to match a *final* segment of a line. The construction *^entire RE\$* constrains the entire RE to match the entire line.

The null RE (e.g., */*) is equivalent to the last RE encountered. See also the last paragraph before *FILES* below.

To understand addressing in *ed* it is necessary to know that at any time there is a *current line*. Generally speaking, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. *Addresses* are constructed as follows:

1. The character *.* addresses the current line.
2. The character *\$* addresses the last line of the buffer.
3. A decimal number *n* addresses the *n*-th line of the buffer.
4. *'x* addresses the line marked with the mark name character *x*, which must be a lower-case letter. Lines are marked with the *k* command described below.
5. A RE enclosed by slashes (*/*) addresses the first line found by searching *forward* from the line *following* the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched. See also the last paragraph before *FILES* below.
6. A RE enclosed in question marks (?) addresses the first line found by searching *backward* from the line *preceding* the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. See also the last paragraph before *FILES* below.
7. An address followed by a plus sign (+) or a minus sign (-) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. The plus sign may be omitted.
8. If an address begins with + or -, the addition or subtraction is taken with respect to the current line; e.g., -5 is understood to mean *.-5*.
9. If an address ends with + or -, then 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of rule 8 immediately above, the address - refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the character ^ in addresses is entirely equivalent to -.) Moreover, trailing + and - characters have a cumulative effect, so *---* refers to the current line less 2.





10. For convenience, a comma (,) stands for the address pair 1,3, while a semicolon (;) stands for the pair ..3.

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (,). They may also be separated by a semicolon (;). In the latter case, the current line (.) is set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches (see rules 5. and 6. above). The second address of any two-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of *ed* commands, the default addresses are shown in parentheses. The parentheses are *not* part of the address; they show that the given addresses are the default.

It is generally illegal for more than one command to appear on a line. However, any command (except *e*, *f*, *r*, or *w*) may be suffixed by *p* or by *l*, in which case the current line is either printed or listed, respectively, as discussed below under the *p* and *l* commands.

(.)a  
<text>

The *append* command reads the given text and appends it after the addressed line; . is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the "appended" text to be placed at the beginning of the buffer.

(.)c  
<text>

The *change* command deletes the addressed lines, then accepts input text that replaces these lines; . is left at the last line input, or, if there were none, at the first line that was not deleted.

(...)d

The *delete* command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

*e file*

The *edit* command causes the entire contents of the buffer to be deleted, and then the named file to be read in; . is set to the last line of the buffer. If no file name is given, the currently-remembered file name, if any, is used (see the *f* command). The number of characters read is typed; *file* is

The first of these is the fact that the  
the first of these is the fact that the  
the first of these is the fact that the  
the first of these is the fact that the  
the first of these is the fact that the

The second of these is the fact that the  
the second of these is the fact that the  
the second of these is the fact that the  
the second of these is the fact that the  
the second of these is the fact that the

The third of these is the fact that the  
the third of these is the fact that the  
the third of these is the fact that the  
the third of these is the fact that the  
the third of these is the fact that the

The fourth of these is the fact that the  
the fourth of these is the fact that the  
the fourth of these is the fact that the  
the fourth of these is the fact that the  
the fourth of these is the fact that the

The fifth of these is the fact that the  
the fifth of these is the fact that the  
the fifth of these is the fact that the  
the fifth of these is the fact that the  
the fifth of these is the fact that the

The sixth of these is the fact that the  
the sixth of these is the fact that the  
the sixth of these is the fact that the  
the sixth of these is the fact that the  
the sixth of these is the fact that the



remembered for possible use as a default file name in subsequent *e*, *r*, and *w* commands. If *file* begins with *!*, the rest of the line is taken to be a shell (*sh*(1)) command whose output is to be read. Such a shell command is *not* remembered as the current file name. See also *DIAGNOSTICS* below.

### **E** *file*

The *Edit* command is like *e*, except that the editor does not check to see if any changes have been made to the buffer since the last *w* command.

### **f** *file*

If *file* is given, the *file-name* command changes the currently-remembered file name to *file*; otherwise, it prints the currently-remembered file name.

### **(1.\$)g/RE/command list**

In the global command, the first step is to mark every line that matches the given RE. Then, for every such line, the given *command list* is executed with *.* initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. All lines of a multi-line list except the last line must be ended with a *\*; *a*, *i*, and *c* commands and associated input are permitted; the *.* terminating input mode may be omitted if it would be the last line of the *command list*. An empty *command list* is equivalent to the *p* command. The *g*, *G*, *v*, and *V* commands are *not* permitted in the *command list*. See also *BUGS* and the last paragraph before *FILES* below.

### **(1.\$)G/RE/**

In the interactive Global command, the first step is to mark every line that matches the given RE. Then, for every such line, that line is printed, *.* is changed to that line, and any *one* command (other than one of the *a*, *c*, *i*, *g*, *G*, *v*, and *V* commands) may be input and is executed. After the execution of that command, the next marked line is printed, and so on; a new-line acts as a null command; an *&* causes the re-execution of the most recent command executed within the current invocation of *G*. Note that the commands input as part of the execution of the *G* command may address and affect *any* lines in the buffer. The *G* command can be terminated by an interrupt signal (ASCII DEL or BREAK).

### **h**

The *help* command gives a short error message that explains the reason for the most recent *?* diagnostic.

### **H**

The *Help* command causes *ed* to enter a mode in which error messages are printed for all subsequent *?* diagnostics. It will also explain the previous *?* if there was one. The *H* command alternately turns this mode on and off; it is initially off.

### **(.)i**

The first part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.

The second part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.

The third part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.

The fourth part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.

The fifth part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.

The sixth part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.

The seventh part of the paper is devoted to a discussion of the general principles of the theory of the structure of the atom. It is shown that the structure of the atom is determined by the laws of quantum mechanics, and that the laws of quantum mechanics are determined by the laws of the theory of the structure of the atom.



<text>

The insert command inserts the given text before the addressed line; . is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the *a* command only in the placement of the input text. Address 0 is not legal for this command.

(...+1)j

The join command joins contiguous lines by removing the appropriate new-line characters. If only one address is given, this command does nothing.

(.)kx

The mark command marks the addressed line with name *x*, which must be a lower-case letter. The address '*x*' then addresses this line; . is unchanged.

(...)l

The list command prints the addressed lines in an unambiguous way: a few non-printing characters (e.g., *tab*, *backspace*) are represented by (hopefully) mnemonic overstrikes, all other non-printing characters are printed in octal, and long lines are folded. An *l* command may be appended to any other command other than *e*, *f*, *r*, or *w*.

(...)ma

The move command repositions the addressed line(s) after the line addressed by *a*. Address 0 is legal for *a* and causes the addressed line(s) to be moved to the beginning of the file; it is an error if address *a* falls within the range of moved lines; . is left at the last line moved.

(...)n

The number command prints the addressed lines, preceding each line by its line number and a tab character; . is left at the last line printed. The *n* command may be appended to any other command other than *e*, *f*, *r*, or *w*.

(...)p

The print command prints the addressed lines; . is left at the last line printed. The *p* command may be appended to any other command other than *e*, *f*, *r*, or *w*; for example, *dp* deletes the current line and prints the new current line.

P

The editor will prompt with a \* for all subsequent commands. The *P* command alternately turns this mode on and off; it is initially off.

q

The quit command causes *ed* to exit. No automatic write of a file is done (but see *DIAGNOSTICS* below).

Q

The editor exits without checking if changes have been made in the buffer since the last *w* command.

The first of these is the fact that the  
... ..  
... ..  
... ..

The second is the fact that the  
... ..  
... ..  
... ..

The third is the fact that the  
... ..  
... ..  
... ..

The fourth is the fact that the  
... ..  
... ..  
... ..  
... ..  
... ..

The fifth is the fact that the  
... ..  
... ..  
... ..  
... ..

The sixth is the fact that the  
... ..  
... ..  
... ..  
... ..

The seventh is the fact that the  
... ..  
... ..  
... ..  
... ..

The eighth is the fact that the  
... ..  
... ..  
... ..

The ninth is the fact that the  
... ..  
... ..  
... ..

The tenth is the fact that the  
... ..  
... ..  
... ..



**(s)r file**

The *r*ead command reads in the given file after the addressed line. If no file name is given, the currently-remembered file name, if any, is used (see *e* and *f* commands). The currently-remembered file name is *not* changed unless *file* is the very first file name mentioned since *ed* was invoked. Address 0 is legal for *r* and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; . is set to the last line read in. If *file* begins with !, the rest of the line is taken to be a shell (*sh*(1)) command whose output is to be read. Such a shell command is *not* remembered as the current file name.

(...)*s*/*RE*/*replacement* /      or  
 (...)*s*/*RE*/*replacement* /*g*

The substitute command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (non-overlapped) matched strings are replaced by the *replacement* if the global replacement indicator *g* appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. It is an error for the substitution to fail on *all* addressed lines. Any character other than space or new-line may be used instead of / to delimit the RE and the *replacement*; . is left at the last line on which a substitution occurred. See also the last paragraph before *FILES* below.

An ampersand (&) appearing in the *replacement* is replaced by the string matching the RE on the current line. The special meaning of & in this context may be suppressed by preceding it by \. As a more general feature, the characters \n, where *n* is a digit, are replaced by the text matched by the *n*-th regular subexpression of the specified RE enclosed between \ ( and \). When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of \ ( starting from the left. When the character % is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The % loses its special meaning when it is in a replacement string of more than one character or is preceded by a \.

A line may be split by substituting a new-line character into it. The new-line in the *replacement* must be escaped by preceding it by \. Such substitution cannot be done as part of a *g* or *v* command list.

**(...)ta**

This command acts just like the *m* command, except that a *copy* of the addressed lines is placed after address *a* (which may be 0); . is left at the last line of the copy.

**u**

The *u*ndo command nullifies the effect of the most recent

The first part of the report is devoted to a description of the work done during the year. It is divided into two main sections, the first of which deals with the work done in the laboratory and the second with the work done in the field. The first section is divided into three parts, the first of which deals with the work done in the laboratory, the second with the work done in the field, and the third with the work done in the laboratory. The second section is divided into two parts, the first of which deals with the work done in the field, and the second with the work done in the laboratory.

The second part of the report is devoted to a description of the results of the work done during the year. It is divided into two main sections, the first of which deals with the results of the work done in the laboratory and the second with the results of the work done in the field. The first section is divided into three parts, the first of which deals with the results of the work done in the laboratory, the second with the results of the work done in the field, and the third with the results of the work done in the laboratory. The second section is divided into two parts, the first of which deals with the results of the work done in the field, and the second with the results of the work done in the laboratory.

The third part of the report is devoted to a description of the conclusions of the work done during the year. It is divided into two main sections, the first of which deals with the conclusions of the work done in the laboratory and the second with the conclusions of the work done in the field. The first section is divided into three parts, the first of which deals with the conclusions of the work done in the laboratory, the second with the conclusions of the work done in the field, and the third with the conclusions of the work done in the laboratory. The second section is divided into two parts, the first of which deals with the conclusions of the work done in the field, and the second with the conclusions of the work done in the laboratory.

The fourth part of the report is devoted to a description of the recommendations of the work done during the year. It is divided into two main sections, the first of which deals with the recommendations of the work done in the laboratory and the second with the recommendations of the work done in the field. The first section is divided into three parts, the first of which deals with the recommendations of the work done in the laboratory, the second with the recommendations of the work done in the field, and the third with the recommendations of the work done in the laboratory. The second section is divided into two parts, the first of which deals with the recommendations of the work done in the field, and the second with the recommendations of the work done in the laboratory.

The fifth part of the report is devoted to a description of the summary of the work done during the year. It is divided into two main sections, the first of which deals with the summary of the work done in the laboratory and the second with the summary of the work done in the field. The first section is divided into three parts, the first of which deals with the summary of the work done in the laboratory, the second with the summary of the work done in the field, and the third with the summary of the work done in the laboratory. The second section is divided into two parts, the first of which deals with the summary of the work done in the field, and the second with the summary of the work done in the laboratory.



command that modified anything in the buffer, namely the most recent *a*, *c*, *d*, *g*, *i*, *j*, *m*, *r*, *s*, *t*, *v*, *G*, or *V* command.

**(1,3)v/RE/command list**

This command is the same as the global command *g* except that the *command list* is executed with *.* initially set to every line that does *not* match the RE.

**(1,3)V/RE/**

This command is the same as the interactive global command *G* except that the lines that are marked during the first step are those that do *not* match the RE.

**(1,3)w file**

The *write* command writes the addressed lines into the named file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless your *umask* setting (see *sh(1)*) dictates otherwise. The currently-remembered file name is *not* changed unless *file* is the very first file name mentioned since *ed* was invoked. If no file name is given, the currently-remembered file name, if any, is used (see *e* and *f* commands); *.* is unchanged. If the command is successful, the number of characters written is typed. If *file* begins with *!*, the rest of the line is taken to be a shell (*sh(1)*) command whose output is to be read. Such a shell command is *not* remembered as the current file name.

**X**

A key string is demanded from the standard input. Subsequent *e*, *r*, and *w* commands will encrypt and decrypt the text with this key by the algorithm of *crypt(1)*. An explicitly empty key turns off encryption.

**(3)=**

The line number of the addressed line is typed; *.* is unchanged by this command.

**!shell command**

The remainder of the line after the *!* is sent to the UNIX shell (*sh(1)*) to be interpreted as a command. Within the text of that command, the unescaped character *%* is replaced with the remembered file name; if a *!* appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, *!!* will repeat the last shell command. If any expansion is performed, the expanded line is echoed; *.* is unchanged.

**(.+1)<new-line>**

An address alone on a line causes the addressed line to be printed. A new-line alone is equivalent to *+.1p*; it is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, *ed* prints a *?* and returns to *its* command level.

Some size limitations: 512 characters per line, 256 characters per global command list, 64 characters per file name, and 128K

177

177

177

The first of these is the fact that the  
population of the country is increasing.

The second is the fact that the

population is increasing at a rapid rate.  
The third is the fact that the population is

increasing at a rapid rate.  
The fourth is the fact that the population is

increasing at a rapid rate.  
The fifth is the fact that the population is  
increasing at a rapid rate.  
The sixth is the fact that the population is  
increasing at a rapid rate.  
The seventh is the fact that the population is  
increasing at a rapid rate.  
The eighth is the fact that the population is  
increasing at a rapid rate.  
The ninth is the fact that the population is  
increasing at a rapid rate.  
The tenth is the fact that the population is  
increasing at a rapid rate.

The eleventh is the fact that the population is  
increasing at a rapid rate.  
The twelfth is the fact that the population is  
increasing at a rapid rate.

The thirteenth is the fact that the population is  
increasing at a rapid rate.

The fourteenth is the fact that the population is  
increasing at a rapid rate.  
The fifteenth is the fact that the population is  
increasing at a rapid rate.  
The sixteenth is the fact that the population is  
increasing at a rapid rate.  
The seventeenth is the fact that the population is  
increasing at a rapid rate.  
The eighteenth is the fact that the population is  
increasing at a rapid rate.

The nineteenth is the fact that the population is  
increasing at a rapid rate.  
The twentieth is the fact that the population is  
increasing at a rapid rate.

The twenty-first is the fact that the population is  
increasing at a rapid rate.

The twenty-second is the fact that the population is  
increasing at a rapid rate.

The twenty-third is the fact that the population is  
increasing at a rapid rate.

The twenty-fourth is the fact that the population is  
increasing at a rapid rate.

The twenty-fifth is the fact that the population is  
increasing at a rapid rate.



characters in the buffer. The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

When reading a file, *ed* discards ASCII NUL characters and all characters after the last new-line. Files (e.g., *a.out*) that contain characters not in the ASCII set (bit 8 on) cannot be edited by *ed*.

If the closing delimiter of a RE or of a replacement string (e.g., */*) would be the last character before a new-line, that delimiter may be omitted, in which case the addressed line is printed. The following pairs of commands are equivalent:

```
s/s1/s2 s/s1/s2/p
g/s1    g/s1/p
?s1     ?s1?
```

## FILES

*/tmp/e#*

temporary; # is the process number.

*ed.hup* work is saved here if the terminal is hung up.

## DIAGNOSTICS

*?* for command errors.

*?file* for an inaccessible file.

(use the *help* and *Help* commands for detailed explanations).

If changes have been made in the buffer since the last *w* command that wrote the entire buffer, *ed* warns the user if an attempt is made to destroy *ed*'s buffer via the *e* or *q* commands: it prints *?* and allows one to continue editing. A second *e* or *q* command at this point will take effect. The *-* command-line option inhibits this feature.

## SEE ALSO

*crypt(1)*, *grep(1)*, *sed(1)*, *sh(1)*.

*A Tutorial Introduction to the UNIX Text Editor* by B. W. Kernighan.

*Advanced Editing on UNIX* by B. W. Kernighan.

## CAVEATS AND BUGS

A *!* command cannot be subject to a *g* or a *v* command.

The *!* command and the *!* escape from the *e*, *r*, and *w* commands cannot be used if the editor is invoked from a restricted shell (see *sh(1)*).

The sequence *\n* in a RE does not match any character.

The *l* command mishandles DEL.

Files encrypted directly with the *crypt(1)* command with the null key cannot be edited.

Because 0 is an illegal address for the *w* command, it is not possible to create an empty file with *ed*.





## NAME

env - set environment for command execution

## SYNOPSIS

env [-] [ name=value ] ... [ command args ]

## DESCRIPTION

*Env* obtains the current *environment*, modifies it according to its arguments, then executes the command with the modified environment. Arguments of the form *name=value* are merged into the inherited environment before the command is executed. The - flag causes the inherited environment to be ignored completely, so that the command is executed with exactly the environment specified by the arguments.

If no command is specified, the resulting environment is printed, one name-value pair per line.

## SEE ALSO

sh(1), exec(2), profile(5), environ(7).

THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

REPORT OF THE CHAIRMAN OF THE COMMITTEE ON THE STUDY OF THE PROBLEM OF THE CHEMICAL INDUSTRY IN THE UNITED STATES

CHICAGO, ILLINOIS



## NAME

eqn, neqn, checkeq - format mathematical text for nroff or troff

## SYNOPSIS

```
eqn [ -dxy ] [ -pn ] [ -sn ] [ -fn ] [ files ]
neqn [ -dxy ] [ -pn ] [ -sn ] [ -fn ] [ files ]
checkeq [ files ]
```

## DESCRIPTION

*Eqn* is a *troff*(1) preprocessor for typesetting mathematical text on a Wang Laboratories, Inc. C/A/T phototypesetter, while *neqn* is used for the same purpose with *nroff*(1) on typewriter-like terminals. Usage is almost always:

```
eqn files | troff
neqn files | nroff
```

or equivalent.

If no files are specified, these programs read from the standard input. A line beginning with **.EQ** marks the start of an equation; the end of an equation is marked by a line beginning with **.EN**. Neither of these lines is altered, so they may be defined in macro packages to get centering, numbering, etc. It is also possible to designate two characters as *delimiters*; subsequent text between delimiters is then treated as *eqn* input. Delimiters may be set to characters *x* and *y* with the command-line argument **-dxy** or (more commonly) with **delim xy** between **.EQ** and **.EN**. The left and right delimiters may be the same character; the dollar sign is often used as such a delimiter. Delimiters are turned off by **delim off**. All text that is neither between delimiters nor between **.EQ** and **.EN** is passed through untouched.

The program *checkeq* reports missing or unbalanced delimiters and **.EQ/.EN** pairs.

Tokens within *eqn* are separated by spaces, tabs, new-lines, braces, double quotes, tildes, and circumflexes. Braces **{ }** are used for grouping; generally speaking, anywhere a single character such as *x* could appear, a complicated construction enclosed in braces may be used instead. Tilde (**~**) represents a full space in the output, circumflex (**^**) half as much.

Subscripts and superscripts are produced with the keywords **sub** and **sup**. Thus *x sub j* makes  $x_j$ , *a sub k sup 2* produces  $a_k^2$ , while  $e^{x^2+y^2}$  is made with *e sup {x sup 2 + y sup 2}*. Fractions are made with **over**: *a over b* yields  $\frac{a}{b}$ ; **sqrt** makes square roots: *1 over sqrt {ax sup 2+bx+c}* results in  $\frac{1}{\sqrt{ax^2+bx+c}}$ .

The keywords **from** and **to** introduce lower and upper limits:  $\lim_{n \rightarrow \infty} \sum_0^n x_i$  is made with *lim from {n -> inf} sum from 0 to n x sub i*. Left and right brackets, braces, etc., of the right height are made with **left** and **right**:

*left [ x sup 2 + y sup 2 over alpha right ] ~ = ~ 1* produces  $\left[ x^2 + \frac{y^2}{\alpha} \right] = 1$ .

Legal characters after **left** and **right** are braces, brackets, bars, **c** and **f**





for ceiling and floor, and "" for nothing at all (useful for a right-side-only bracket). A left *thing* need not have a matching right *thing*.

Vertical piles of things are made with *pile*, *lpile*, *cpile*, and *rpile*:

*pile {a above b above c}* produces  $\begin{smallmatrix} a \\ b \\ c \end{smallmatrix}$ . Piles may have arbitrary numbers of elements; *lpile* left-justifies, *pile* and *cpile* center (but with different vertical spacing), and *rpile* right justifies. Matrices are made with *matrix*:

*matrix { lcol { x sub i above y sub 2 } ccol { 1 above 2 } }* produces  $\begin{smallmatrix} x_i & 1 \\ y_2 & 2 \end{smallmatrix}$ .

In addition, there is *rcol* for a right-justified column.

Diacritical marks are made with *dot*, *dotdot*, *hat*, *tilde*, *bar*, *vec*, *dyad*, and *under*. *x dot* =  $f(t)$  *bar* is  $\bar{x} = \overline{f(t)}$ , *y dotdot bar*  $\sim = \sim n$  *under* is  $\bar{y} = \underline{n}$ , and *x vec*  $\sim = \sim y$  *dyad* is  $\vec{x} = \vec{y}$ .

Point sizes and fonts can be changed with *size n* or *size ±n*, *roman*, *italic*, *bold*, and *font n*. Point sizes and fonts can be changed globally in a document by *gsize n* and *gfont n*, or by the command-line arguments *-sn* and *-fn*.

Normally, subscripts and superscripts are reduced by 3 points from the previous size; this may be changed by the command-line argument *-pn*.

Successive display arguments can be lined up. Place *mark* before the desired lineup point in the first equation; place *lineup* at the place that is to line up vertically in subsequent equations.

Shorthands may be defined or existing keywords redefined with *define*:

*define thing % replacement %*

defines a new token called *thing* that will be replaced by *replacement* whenever it appears thereafter. The % may be any character that does not occur in *replacement*.

Keywords such as *sum* ( $\Sigma$ ), *int* ( $\int$ ), *inf* ( $=$ ), and shorthands such as  $\geq$  ( $\geq$ ),  $\neq$  ( $\neq$ ), and  $\rightarrow$  ( $\rightarrow$ ) are recognized. Greek letters are spelled out in the desired case, as in *alpha* ( $\alpha$ ), or *GAMMA* ( $\Gamma$ ). Mathematical words such as *sin*, *cos*, and *log* are made Roman automatically. *Troff*(1) four-character escapes such as  $\backslash(dd$  ( $\dagger$ ) and  $\backslash(bs$  ( $\textcircled{b}$ ) may be used anywhere. Strings enclosed in double quotes ("...") are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with *troff*(1) when all else fails. Full details are given in the manual cited below.

#### SEE ALSO

*Typesetting Mathematics—User's Guide* by B. W. Kernighan and L. L. Cherry.

*New Graphic Symbols for EQN and NEQN* by C. Scrocca.

*mm*(1), *mmt*(1), *tbl*(1), *troff* (1), *eqnchar*(7), *mm*(7), *mv*(7).

#### BUGS

To embolden digits, parentheses, etc., it is necessary to quote them, as in bold "12.3".

See also *BUGS* under *troff*(1).

The first part of the book is devoted to a general introduction to the subject of the history of the world. It is a very interesting and well-written book, and it is a very good introduction to the subject of the history of the world. It is a very good introduction to the subject of the history of the world.

The second part of the book is devoted to a general introduction to the subject of the history of the world. It is a very interesting and well-written book, and it is a very good introduction to the subject of the history of the world. It is a very good introduction to the subject of the history of the world.

The third part of the book is devoted to a general introduction to the subject of the history of the world. It is a very interesting and well-written book, and it is a very good introduction to the subject of the history of the world. It is a very good introduction to the subject of the history of the world.

The fourth part of the book is devoted to a general introduction to the subject of the history of the world. It is a very interesting and well-written book, and it is a very good introduction to the subject of the history of the world. It is a very good introduction to the subject of the history of the world.

The fifth part of the book is devoted to a general introduction to the subject of the history of the world. It is a very interesting and well-written book, and it is a very good introduction to the subject of the history of the world. It is a very good introduction to the subject of the history of the world.



## NAME

f77 - Fortran 77 compiler

## SYNOPSIS

f77 [ option ] ... file ...

## DESCRIPTION

F77 is the UNIX Fortran 77 compiler. It accepts several types of arguments:

Arguments whose names end with '.f' are taken to be Fortran 77 source programs; they are compiled, and each object program is left on the file in the current directory whose name is that of the source with '.o' substituted for '.f'.

Arguments whose names end with '.r' or '.e' are taken to be Ratfor or EFL source programs, respectively; these are first transformed by the appropriate preprocessor, then compiled by f77.

In the same way, arguments whose names end with '.c' or '.s' are taken to be C or assembly source programs and are compiled or assembled, producing a '.o' file.

Floating point is implemented with the Motorola Fast Floating Point Package. The compiler will accept double precision declarations, but handles them silently as single precision.

The following options have the same meaning as in cc(1). See ld(1) for load-time options.

- c Suppress loading and produce '.o' files for each source file.
- p Prepare object files for profiling, see *prof*(1).
- S Compile the named programs, and leave the assembler-listing on corresponding files suffixed '.s'.
- o output  
Name the final output file *output* instead of 'a.out'.

The following options are peculiar to f77.

- onetrip  
Compile DO loops that are performed at least once if reached. (Fortran 77 DO loops are not performed at all if the upper limit is smaller than the lower limit.)
- 1 Same as '-onetrip'.
- U Do not convert UPPERCASE to lowercase.
- u Make the default type of a variable 'undefined' rather than using the default Fortran rules.
- C Compile code to check that subscripts are within declared array bounds.
- w Suppress all warning messages. If the option is '-w66', only Fortran 66 compatibility warnings are suppressed.
- Nx Change default sizes of certain tables. 'x' is one of the letters q,x,s,c or n followed directly by a decimal number. The defaults are:

1. The first part of the document discusses the importance of maintaining accurate records of all transactions and activities. It emphasizes the need for transparency and accountability in all financial dealings.

2. The second part of the document outlines the specific procedures and protocols that must be followed when conducting financial transactions. This includes the use of standardized forms and the requirement for proper authorization and documentation.

3. The third part of the document provides a detailed overview of the various financial systems and tools that are used to manage and track transactions. It describes the functionality of each system and how they are integrated into the overall financial management process.

4. The fourth part of the document discusses the role of the finance department in ensuring the accuracy and integrity of the financial data. It highlights the importance of regular audits and the implementation of internal controls to prevent errors and fraud.

5. The fifth part of the document provides a summary of the key findings and recommendations from the review. It identifies areas for improvement and provides specific suggestions for how the finance department can enhance its operations and reporting.

6. The sixth part of the document contains a list of references and sources used in the review. This includes various financial reports, industry standards, and academic research on financial management practices.

7. The seventh part of the document provides a detailed description of the financial data and transactions that were analyzed during the review. This includes a breakdown of the data by department, project, and time period.

8. The eighth part of the document discusses the results of the analysis and the impact of the findings on the organization's financial performance. It provides a clear and concise summary of the key findings and their implications for the future.

9. The ninth part of the document provides a list of recommendations and action items that must be implemented to address the identified issues. It includes a timeline for the implementation of these recommendations and a responsible party for each item.

10. The tenth part of the document provides a final summary and conclusion. It reiterates the importance of maintaining accurate financial records and the need for continuous improvement in financial management practices.



- Nq150 Maximum number of equivalence statements
- Nx200 Maximum number of external definitions
- Ns701 Maximum number of statement labels
- Nc20 Maximum nesting of control structures
- Nn401 Size of hash table
- l2 Use short integers (2 bytes) for Fortran type integer.
- l4 Use long integers (4 bytes) for Fortran type integer (default). Additionally, make subscripts long.
- ls Make subscripts short (default).
- F Apply EFL and Ratfor preprocessor to relevant files, put the result in the file with the suffix changed to '.f', but do not compile.
- m Apply the M4 preprocessor to each '.r' or '.e' file before transforming it with the Ratfor or EFL preprocessor.
- Ex Use the string *x* as an EFL option in processing '.e' files.
- Rx Use the string *x* as a Ratfor option in processing '.r' files.

Other arguments are taken to be either loader option arguments, or F77-compatible object programs, typically produced by an earlier run, or perhaps libraries of F77-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with name 'a.out'.

#### FILES

file.[fresc]	input file
file.o	object file
a.out	loaded output
/usr/lib/f77pass1	compiler
/lib/c1	pass 2
/lib/c2	pass3
/usr/lib/libF77.a	intrinsic function library
/usr/lib/libI77.a	Fortran I/O library
/lib/libc.a	C library, see section 3
/lib/libm.a	Mathematical library
/lib/libffp.a	Floating point library

#### SEE ALSO

S. I. Feldman, P. J. Weinberger, *A Portable Fortran 77 Compiler*  
prof(1), cc(1), ld(1)

#### DIAGNOSTICS

The diagnostics produced by *f77* itself are intended to be self-explanatory. Occasional messages may be produced by the loader.

#### BUGS

This compiler has run the Fortran Compiler Validation Suite FCVS 78 from the Federal Cobol Compiler Test Service (FCCTS). FCVS 78 is also known as "navytest". It passed all tests except two. One error is due to floating point accuracy. The other error is that a format statement must be given before an "assign"-statement that assigns the label of the format statement to an integer variable.





## NAME

find - find files

## SYNOPSIS

find path-name-list expression

## DESCRIPTION

*Find* recursively descends the directory hierarchy for each path name in the *path-name-list* (i.e., one or more path names) seeking files that match a boolean *expression* written in the primaries given below. In the descriptions, the argument *n* is used as a decimal integer where *+n* means more than *n*, *-n* means less than *n* and *n* means exactly *n*.

- name *file*** True if *file* matches the current file name. Normal shell argument syntax may be used if escaped (watch out for [, ? and \*).
- perm *onum*** True if the file permission flags exactly match the octal number *onum* (see *chmod(1)*). If *onum* is prefixed by a minus sign, more flag bits (017777, see *stat(2)*) become significant and the flags are compared:  
(flags&onum)==onum
- type *c*** True if the type of the file is *c*, where *c* is b, c, d, p, or f for block special file, character special file, directory, fifo (a.k.a named pipe), or plain file.
- links *n*** True if the file has *n* links.
- user *uname*** True if the file belongs to the user *uname*. If *uname* is numeric and does not appear as a login name in the */etc/passwd* file, it is taken as a user ID.
- group *gname*** True if the file belongs to the group *gname*. If *gname* is numeric and does not appear in the */etc/group* file, it is taken as a group ID.
- size *n*** True if the file is *n* blocks long (512 bytes per block).
- atime *n*** True if the file has been accessed in *n* days.
- mtime *n*** True if the file has been modified in *n* days.
- ctime *n*** True if the file has been changed in *n* days.
- exec *cmd*** True if the executed *cmd* returns a zero value as exit status. The end of *cmd* must be punctuated by an escaped semicolon. A command argument {} is replaced by the current path name.
- ok *cmd*** Like **-exec** except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing y.
- print** Always true; causes the current path name to be printed.
- cpio *device*** Write the current file on *device* in *cpio* (5) format (5120 byte records).

1. The first part of the document is a letter from the President of the United States to the Congress.

2. The second part of the document is a report from the Secretary of the Department of the Interior.

3. The third part of the document is a report from the Secretary of the Department of the Treasury.

4. The fourth part of the document is a report from the Secretary of the Department of the Army.

5. The fifth part of the document is a report from the Secretary of the Department of the Navy.

6. The sixth part of the document is a report from the Secretary of the Department of the Air Force.

7. The seventh part of the document is a report from the Secretary of the Department of the Coast Guard.

8. The eighth part of the document is a report from the Secretary of the Department of the Marine Corps.

9. The ninth part of the document is a report from the Secretary of the Department of the Army.

10. The tenth part of the document is a report from the Secretary of the Department of the Navy.



**-newer *file*** True if the current file has been modified more recently than the argument *file*.

**( *expression* )** True if the parenthesized expression is true (parentheses are special to the shell and must be escaped).

The primaries may be combined using the following operators (in order of decreasing precedence):

- 1) The negation of a primary (! is the unary *not* operator).
- 2) Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries).
- 3) Alternation of primaries (-o is the *or* operator).

#### EXAMPLE

To remove all files named *a.out* or *\*.o* that have not been accessed for a week:

```
find / \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

#### FILES

/etc/passwd, /etc/group

#### SEE ALSO

cpio(1), sh(1), test(1), stat(2), cpio(5), fs(5).

1. The first part of the document is a letter from the President of the United States to the Congress, dated January 1, 1800.

2. The second part is a report from the Secretary of the Treasury, dated January 1, 1800.

3. The third part is a report from the Secretary of the Navy, dated January 1, 1800.

4. The fourth part is a report from the Secretary of the War, dated January 1, 1800.

5. The fifth part is a report from the Secretary of the Interior, dated January 1, 1800.

6. The sixth part is a report from the Secretary of the State, dated January 1, 1800.

7. The seventh part is a report from the Secretary of the War, dated January 1, 1800.

8. The eighth part is a report from the Secretary of the Navy, dated January 1, 1800.

9. The ninth part is a report from the Secretary of the Treasury, dated January 1, 1800.

10. The tenth part is a report from the Secretary of the State, dated January 1, 1800.



## NAME

*fsdb* - file system debugger

## SYNOPSIS

*/etc/fsdb* special [ - ]

## DESCRIPTION

*Fsdb* can be used to patch up a damaged file system after a crash. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an i-node. These greatly simplify the process of correcting control block entries or descending the file system tree.

*Fsdb* contains several error checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking *fsdb* with the optional - argument or by the use of the O symbol. (*Fsdb* reads the i-size and f-size entries from the superblock of the file system as the basis for these checks.)

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

*Fsdb* reads a block at a time and will therefore work with raw as well as block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The symbols recognized by *fsdb* are:

#	absolute address
i	convert from i-number to i-node address
b	convert to block address
d	directory slot offset
+, -	address arithmetic
q	quit
>, <	save, restore an address
=	numerical assignment
=+	incremental assignment
=-	decremental assignment
="	character string assignment
O	error checking flip flop
p	general print facilities
f	file print facility
B	byte mode
W	word mode
D	double word mode
!	escape to shell

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the delete character. If a number follows the p symbol, that many entries are printed. A check is made to detect block boundary overflows

1907-1908

1907-1908

The first of the year was a very successful one for the school. The students showed a great interest in their studies and the teachers were able to give them a thorough education. The school was well attended and the students were very happy.

The second of the year was also a very successful one. The students showed a great interest in their studies and the teachers were able to give them a thorough education. The school was well attended and the students were very happy.

The third of the year was also a very successful one. The students showed a great interest in their studies and the teachers were able to give them a thorough education. The school was well attended and the students were very happy.

The fourth of the year was also a very successful one. The students showed a great interest in their studies and the teachers were able to give them a thorough education. The school was well attended and the students were very happy.

1907-1908	
1907-1908	1
1907-1908	2
1907-1908	3
1907-1908	4
1907-1908	5
1907-1908	6
1907-1908	7
1907-1908	8
1907-1908	9
1907-1908	10
1907-1908	11
1907-1908	12
1907-1908	13
1907-1908	14
1907-1908	15
1907-1908	16
1907-1908	17
1907-1908	18
1907-1908	19
1907-1908	20
1907-1908	21
1907-1908	22
1907-1908	23
1907-1908	24
1907-1908	25
1907-1908	26
1907-1908	27
1907-1908	28
1907-1908	29
1907-1908	30
1907-1908	31
1907-1908	32
1907-1908	33
1907-1908	34
1907-1908	35
1907-1908	36
1907-1908	37
1907-1908	38
1907-1908	39
1907-1908	40
1907-1908	41
1907-1908	42
1907-1908	43
1907-1908	44
1907-1908	45
1907-1908	46
1907-1908	47
1907-1908	48
1907-1908	49
1907-1908	50
1907-1908	51
1907-1908	52
1907-1908	53
1907-1908	54
1907-1908	55
1907-1908	56
1907-1908	57
1907-1908	58
1907-1908	59
1907-1908	60
1907-1908	61
1907-1908	62
1907-1908	63
1907-1908	64
1907-1908	65
1907-1908	66
1907-1908	67
1907-1908	68
1907-1908	69
1907-1908	70
1907-1908	71
1907-1908	72
1907-1908	73
1907-1908	74
1907-1908	75
1907-1908	76
1907-1908	77
1907-1908	78
1907-1908	79
1907-1908	80
1907-1908	81
1907-1908	82
1907-1908	83
1907-1908	84
1907-1908	85
1907-1908	86
1907-1908	87
1907-1908	88
1907-1908	89
1907-1908	90
1907-1908	91
1907-1908	92
1907-1908	93
1907-1908	94
1907-1908	95
1907-1908	96
1907-1908	97
1907-1908	98
1907-1908	99
1907-1908	100

The fifth of the year was also a very successful one. The students showed a great interest in their studies and the teachers were able to give them a thorough education. The school was well attended and the students were very happy.



since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed. The print options available are:

i	print as i-nodes
d	print as directories
o	print as octal words
e	print as decimal words
c	print as characters
b	print as octal bytes

The **f** symbol is used to print data blocks associated with the current i-node. If followed by a number, that block of the file is printed. (Blocks are numbered from zero.) The desired print option letter follows the block number, if present, or the **f** symbol. This print facility works for small as well as large files. It checks for special devices and that the block pointers used to find the data are not zero.

Dots, tabs and spaces may be used as function delimiters but are not necessary. A line with just a new-line character will increment the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing the user to step through a region of a file system. Information is printed in a format appropriate to the data type. Bytes, words and double words are displayed with the octal address followed by the value in octal and decimal. A **B** or **D** is appended to the address for byte and double word values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. Inodes are printed with labeled fields describing each element.

The following mnemonics are used for i-node examination and refer to the current working i-node:

md	mode
ln	link count
uid	user ID number
gid	group ID number
s0	high byte of file size
s1	low word of file size
a#	data block numbers (0 - 12)
at	access time
mt	modification time
maj	major device number
min	minor device number

#### EXAMPLES

386i	prints i-number 386 in an i-node format. This now becomes the current working i-node.
ln=4	changes the link count for the working i-node to 4.
ln+=1	increments the link count by 1.
fc	prints, in ASCII, block zero of the file associated with the working i-node.





- 2i.fd            prints the first 32 directory entries for the root i-node of this file system.
- d5i.fc           changes the current i-node to that associated with the 5th directory entry (numbered from zero) found from the above command. The first 512 bytes of the file are then printed in ASCII.
- 1b.p0o          prints the superblock of this file system in octal.
- 2i.a0b.d7=3     changes the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.
- d7.nm="name"    changes the name field in the directory slot to the given string. Quotes are optional when used with nm if the first character is alphabetic.

**SEE ALSO**

fsck(1M), dir(5), fs(5).

1. The first part of the document is a list of the names of the persons who were present at the meeting.

1. 2

2. The second part of the document is a list of the names of the persons who were present at the meeting.

2. 3

3. The third part of the document is a list of the names of the persons who were present at the meeting.

3. 4

4. The fourth part of the document is a list of the names of the persons who were present at the meeting.

4. 5

Page 10

Page 10

Page 10



## NAME

getopt — parse command options

## SYNOPSIS

set — `getopt optstring \$\*

## DESCRIPTION

*Getopt* is used to break up options in command lines for easy parsing by shell procedures, and to check for legal options. *Optstring* is a string of recognized option letters (see *getopt(3C)*); if a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by white space. The special option — is used to delimit the end of the options. *Getopt* will place — in the arguments at the end of the options, or recognize it if used explicitly. The shell arguments (\$1 \$2 . . .) are reset so that each option is preceded by a — and in its own shell argument; each option argument is also in its own shell argument.

## DIAGNOSTICS

*Getopt* prints an error message on the standard error when it encounters an option letter not included in *optstring*.

## EXAMPLE

The following code fragment shows how one might process the arguments for a command that can take the options a and b, and the option o, which requires an argument.

```
set — `getopt abo: $*`
if [ $? != 0 ]
then
    echo $USAGE
    exit 2
fi
for i in $*
do
    case $i in
        -a | -b)    FLAG=$i; shift;;
        -o)        OARG=$2; shift; shift;;
        —)        shift; break;;
    esac
done
```

This code will accept any of the following as equivalent:

```
cmd -aoarg file file
cmd -a -o arg file file
cmd -oarg -a file file
cmd -a -oarg — file file
```

## SEE ALSO

sh(1), *getopt(3C)*.

...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...



## NAME

hyphen - find hyphenated words

## SYNOPSIS

hyphen files

## DESCRIPTION

*Hyphen* finds all the hyphenated words in *files* and prints them on the standard output. If no arguments are given, the standard input is used. Thus *hyphen* may be used as a filter.

## BUGS

*Hyphen* can't cope with hyphenated *italic* (i.e., underlined) words; it will often miss them completely, or mangle them.

*Hyphen* occasionally gets confused, but with no ill effects other than spurious extra output.





**NAME**

id - print user and group IDs and names

**SYNOPSIS**

id

**DESCRIPTION**

*Id* writes a message on the standard output giving the user and group IDs and the corresponding names of the invoking process. If the effective and real IDs do not match, both are printed.

**SEE ALSO**

logname(1), getuid(2), getgid(2).

1. The first part of the document is a list of the names of the persons who were present at the meeting.

2. The second part of the document is a list of the names of the persons who were present at the meeting.

3. The third part of the document is a list of the names of the persons who were present at the meeting.



## NAME

install - install commands

## SYNOPSIS

```
install [ -c dira ] [ -f dirb ] [ -i ] [ -n dirc ] [ -o ] [ -s ] file [
dirx ... ]
```

## DESCRIPTION

*install* is a command most commonly used in "makefiles" (see *make(1)*) to install a *file* (updated target file) in a specific place within a file system. Each *file* is installed by copying it into the appropriate directory, thereby retaining the mode and owner of the original command. The program prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* ...) are given, *install* will search (using *find(1)*) a set of default directories (*/bin*, */usr/bin*, */etc*, */lib*, and */usr/lib*, in that order) for a file with the same name as *file*. When the first occurrence is found, *install* issues a message saying that it is overwriting that file with *file*, and proceeds to do so. If the file is not found, the program states this and exits without further action.

If one or more directories (*dirx* ...) are specified after *file*, those directories will be searched before the directories specified in the default list.

The meanings of the options are:

- c *dira*** Installs a new command in the directory specified in *dira*. Looks for *file* in *dira* and installs it there if it is not found. If it is found, *install* issues a message saying that the file already exists, and exits without overwriting it. May be used alone or with the **-s** option.
- f *dirb*** Forces *file* to be installed in given directory, whether or not one already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to 755 and *bin*, respectively. If the file already exists, the mode and owner will be that of the already existing file. May be used alone or with the **-o** or **-s** options.
- i** Ignores default directory list, searching only through the given directories (*dirx* ...). May be used alone or with any other options other than **-c** and **-f**.
- n *dirc*** If *file* is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file will be set to 755 and *bin*, respectively. May be used alone or with any other options other than **-c** and **-f**.
- o** If *file* is found, this option saves the "found" file by copying it to *OLDfile* in the directory in which it was found. May be used alone or with any other options other than **-c**.





-s

Suppresses printing of messages other than error messages. May be used alone or with any other options.

SEE ALSO  
mk(8).

and the other members of the family  
and the other members of the family





## NAME

ld - loader

## SYNOPSIS

ld [ option ] file ...

## DESCRIPTION

*Ld* combines several object programs into one, resolves external references, and searches libraries. In the simplest case several object *files* are given, and *ld* combines them, producing an object module which can be either executed or become the input for a further *ld* run. (In the latter case, the *-r* option must be given to preserve the relocation bits.) The output of *ld* is left on *a.out*. This file is made executable only if no errors occurred during the load.

The argument routines are concatenated in the order specified. The entry point of the output is the beginning of the first routine.

If any argument is a library, it is searched exactly once at the point it is encountered in the argument list. Only those routines defining an unresolved external reference are loaded. If a routine from a library references another routine in the library, and the library has not been processed by *ranlib*(1), the referenced routine must appear after the referencing routine in the library. Thus the order of programs within libraries may be important. If the first member of a library is named *'\_\_SYMDEF'*, then it is understood to be a dictionary for the library such as produced by *ranlib*; the dictionary is searched iteratively to satisfy as many references as possible.

The symbols *'\_etext'*, *'\_edata'* and *'\_end'* (*'etext'*, *'edata'* and *'end'* in C) are reserved, and if referred to, are set to the first location above the program, the first location above initialized data, and the first location above all data respectively. It is erroneous to define these symbols.

*Ld* understands several options. Except for *-l*, they should appear before the file names.

- s* 'Strip' the output, that is, remove the symbol table and relocation bits to save space (but impair the usefulness of the debugger). This information can also be removed by *strip*(1).
- u* Take the following argument as a symbol and enter it as undefined in the symbol table. This is useful for loading wholly from a library, since initially the symbol table is empty and an unresolved reference is needed to force the loading of the first routine.
- lx* This option is an abbreviation for the library name *'/lib/libx.a'*, where *x* is a string. If that does not exist, *ld* tries *'/usr/lib/libx.a'*. A library is searched when its name is encountered, so the placement of a *-l* is significant.
- 4* This option changes the library name *'/lib/libx.a'* to *'/lib/libLx.a'*. For example instead of */lib/libc.a* the library */lib/libLc.a* is searched. These libraries are assumed to include modules which have been produced with *cc -4*.
- x* Do not preserve local (non-globl) symbols in the output symbol table; only enter external symbols. This option saves some space





in the output file.

- r Generate relocation bits in the output file so that it can be the subject of another *ld* run. This flag also prevents final definitions from being given to common symbols, and suppresses the 'undefined symbol' diagnostics.
- d Force definition of common storage even if the -r flag is present.
- o The *name* argument after -o is used as the name of the *ld* output file, instead of *a.out*.
- p This argument produces a symbol listing on stdout.
- y This argument links objects for an unmapped address range, such that code, data and bss are contiguous, starting from 0. Useful for Macsbug etc.
- b This option is used to link unix or standalone programs, where code is in segment 0 and data and bss in segment 1. If neither -b nor -y are given, normal user-mode programs loadable by the kernel are produced, with code in segment 6 and data and bss in segment 8.
- m This option produces "S-records" for Macsbug on file *dm.out*.
- e The following argument is taken to be the name of the entry point of the loaded program; location 0 is the default.
- O This is an overlay file, only the text segment will be replaced by *ezec(2)*. Shared data must have the same layout as in the program overlaid.
- D The next argument is a decimal number that sets the size of the data segment.

#### FILES

/lib/lib*.a	libraries
/lib/libL*.a	libraries with four-byte integers
/usr/lib/lib*.a	more libraries
a.out	output file

#### SEE ALSO

as(1), ar(1), cc(1), ranlib(1)

#### BUGS

Overlays are not tested.

at the time of the meeting, the committee was not in a position to make a final decision on the matter. The committee will continue to work on the matter and will report back to the assembly at the next meeting.

The committee will also be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.

The committee will be holding a meeting on the 15th of next month. This meeting will be open to the public and will be held in the main hall of the town hall.

The committee will be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.

The committee will be holding a meeting on the 15th of next month. This meeting will be open to the public and will be held in the main hall of the town hall.

The committee will be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.

The committee will be holding a meeting on the 15th of next month. This meeting will be open to the public and will be held in the main hall of the town hall.

The committee will be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.

The committee will be holding a meeting on the 15th of next month. This meeting will be open to the public and will be held in the main hall of the town hall.

The committee will be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.

The committee will be holding a meeting on the 15th of next month. This meeting will be open to the public and will be held in the main hall of the town hall.

The committee will be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.

The committee will be holding a meeting on the 15th of next month. This meeting will be open to the public and will be held in the main hall of the town hall.

The committee will be looking at the possibility of holding a public hearing on the matter. This will allow the public to express their views on the matter and will also allow the committee to hear from the public.



**NAME**

line - read one line

**SYNOPSIS**

line

**DESCRIPTION**

*Line* copies one line (up to a new-line) from the standard input and writes it on the standard output. It returns an exit code of 1 on EOF and always prints at least a new-line. It is often used within shell files to read from the user's terminal.

**SEE ALSO**

sh(1), read(2).

1900

1901

1902

1903

1904

1905

The following table shows the results of the experiments conducted during the year 1900. The first column gives the date of the experiment, the second column the name of the person who conducted it, and the third column the results obtained. The results are given in the form of a table, the columns of which are headed by the names of the persons who conducted the experiments. The results are given in the form of a table, the columns of which are headed by the names of the persons who conducted the experiments.

1906



1907

1908



**NAME**

logname — get login name

**SYNOPSIS**

logname

**DESCRIPTION**

*Logname* returns the contents of the environment variable **\$LOGNAME**, which is set when a user logs into the system.

**FILES**

/etc/profile

**SEE ALSO**

env(1), login(1), logname(3X), environ(7).





- F cause directories to be marked with a trailing '/' and executable files to be marked with a trailing '\*'; this is the default if the last character of the name the program is invoked with is a 'f'.
- R recursively list subdirectories encountered.

The mode printed under the -l option contains 11 characters which are interpreted as follows: the first character is

- d if the entry is a directory;
- b if the entry is a block-type special file;
- c if the entry is a character-type special file;
- m if the entry is a multiplexor-type character special file;
- if the entry is a plain file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next to permissions to others in the same user-group; and the last to all others. Within each set the three characters indicate permission respectively to read, to write, to execute the file as a program. For a directory, 'execute' permission is interpreted to mean permission to search the directory for a specified file. The permissions are indicated as follows:

- r if the file is readable;
- w if the file is writable;
- x if the file is executable;
- if the indicated permission is not granted.

The group-execute permission character is given as s if the file has set-group-ID mode; likewise the user-execute permission character is given as S if the file has set-user-ID mode.

The last character of the mode (normally 'x' or '-') is t if the 1000 bit of the mode is on. See *chmod(1)* for the meaning of this mode.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks is printed.

FILES

/etc/passwd to get user ID's for 'ls -l'.  
 /etc/group to get group ID's for 'ls -g'.

FILES

Newline and tab are considered printing characters in file names.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as "ls -s" is much different than "ls -s | lpr". On the other hand, not doing this setting would make old shell scripts which used *ls* almost certain losers.

Column widths choices are poor for terminals which can tab.

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE

THE JOURNAL OF THE  
THE JOURNAL OF THE



## NAME

**ls** — list contents of directory

## SYNOPSIS

**ls** [ -abcdgilmqrstuxlCFR ] name ...

**l** [ **ls** options ] name ...

## DESCRIPTION

For each directory argument, **ls** lists the contents of the directory; for each file argument, **ls** repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The format chosen depends on whether the output is going to a teletype, and may also be controlled by option flags. The default format for a teletype is to list the contents of directories in multi-column format, with the entries sorted down the columns. (Files which are not the contents of a directory being interpreted are always sorted across the page rather than down the page in columns. This is because the individual file names may be arbitrarily long.) If the standard output is not a teletype, the default format is to list one entry per line. Finally, there is a stream output format in which files are listed across the page, separated by '.' characters. The **-m** flag enables this format; when invoked as **/** this format is also used.

There are an unbelievable number of options:

- l** List in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file. (See below.) If the file is a special file the size field will instead contain the major and minor device numbers.
- t** Sort by time modified (latest first) instead of by name, as is normal.
- a** List all entries; usually '.' and '..' are suppressed.
- s** Give size in blocks, including indirect blocks, for each entry.
- d** If argument is a directory, list only its name, not its contents (mostly used with **-l** to get status on directory).
- r** Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.
- u** Use time of last access instead of last modification for sorting (**-t**) or printing (**-l**).
- c** Use time of file creation for sorting or printing.
- i** Print i-number in first column of the report for each file listed.
- f** Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off **-l**, **-t**, **-s**, and **-r**, and turns on **-a**; the order is the order in which entries appear in the directory.
- g** Give group ID instead of owner ID in long listing.
- m** force stream output format
- l** force one entry per line output format, e.g. to a teletype
- C** force multi-column output, e.g. to a file or a pipe
- q** force printing of non-graphic characters in file names as the character '?'; this normally happens only if the output device is a teletype
- b** force printing of non-graphic characters to be in the \ddd notation, in octal.
- x** force columnar printing to be sorted across rather than down the page; this is the default if the last character of the name the program is invoked with is an 'x'.





## NAME

m4 — macro processor

## SYNOPSIS

m4 [ options ] [ files ]

## DESCRIPTION

M4 is a macro processor intended as a front end for Ratfor, C, and other languages. Each of the argument files is processed in order; if there are no files, or if a file name is —, the standard input is read. The processed text is written on the standard output.

The options and their effects are as follows:

- e Operate interactively. Interrupts are ignored and the output is unbuffered. Using this mode requires a special state of mind.
- s Enable line sync output for the C preprocessor (#line ...)
- Bint Change the size of the push-back and argument collection buffers from the default of 4,096.
- Hint Change the size of the symbol table hash array from the default of 199. The size should be prime.
- Sint Change the size of the call stack from the default of 100 slots. Macros take three slots, and non-macro arguments take one.
- Tint Change the size of the token buffer from the default of 512 bytes.

To be effective, these flags must appear before any file names and before any —D or —U flags:

- Dname[=val]  
Defines *name* to *val* or to null in *val*'s absence.
- Uname  
undefines *name*.

Macro calls have the form:

name(arg1,arg2,..., argn)

The ( must immediately follow the name of the macro. If a defined macro name is not followed by a (, it is deemed to have no arguments. Leading unquoted blanks, tabs, and new-lines are ignored while collecting arguments. Potential macro names consist of alphabetic letters, digits, and underscore \_, where the first character is not a digit.

Left and right single quotes are used to quote strings. The value of a quoted string is the string stripped of the quotes.

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses which happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

M4 makes available the following built-in macros. They may be redefined, but once this is done the original meaning is lost. Their values are null unless otherwise stated.

100 100 100

100 100 100

100 100 100

100 100 100

100 100 100



- define** the second argument is installed as the value of the macro whose name is the first argument. Each occurrence of `$n` in the replacement text, where `n` is a digit, is replaced by the `n`-th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; `$#` is replaced by the number of arguments; `$*` is replaced by a list of all the arguments separated by commas; `$@` is like `$*`, but each argument is quoted (with the current quotes).
- undefine** removes the definition of the macro named in its argument.
- defn** returns the quoted definition of its argument(s). It is useful for renaming macros, especially built-ins.
- pushdef** like *define*, but saves any previous definition.
- popdef** removes current definition of its argument(s), exposing the previous one if any.
- ifdef** if the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is null. The word *unix* is predefined on UNIX versions of *m4*.
- shift** returns all but its first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.
- changequote** change quote symbols to the first and second arguments. The symbols may be up to five characters long. *Changequote* without arguments restores the original values (i.e., `'`).
- changecom** change left and right comment markers from the default `#` and new-line. With no arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes new-line. With two arguments, both markers are affected. Comment markers may be up to five characters long.
- divert** *m4* maintains 10 output streams, numbered 0-9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. The *divert* macro changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded.
- undivert** causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.
- divnum** returns the value of the current output stream.
- dnl** reads and discards characters up to and including the next new-line.





ifelse	has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, null.
incr	returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.
decr	returns the value of its argument decremented by 1.
eval	evaluates its argument as an arithmetic expression, using 32-bit arithmetic. Operators include +, -, *, /, %, ^ (exponentiation), bitwise &,  , ^, and ~; relationals; parentheses. Octal and hex numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.
len	returns the number of characters in its argument.
index	returns the position in its first argument where the second argument begins (zero origin), or -1 if the second argument does not occur.
substr	returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.
translit	transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
include	returns the contents of the file named in the argument.
sinclude	is identical to <i>include</i> , except that it says nothing if the file is inaccessible.
syscmd	executes the UNIX command given in the first argument. No value is returned.
sysval	is the return code from the last call to <i>syscmd</i> .
maketemp	fills in a string of XXXXX in its argument with the current process ID.
m4exit	causes immediate exit from <i>m4</i> . Argument 1, if given, is the exit code; the default is 0.
m4wrap	argument 1 will be pushed back at final EOF; example: <i>m4wrap(`cleanup(`)</i>
errprint	prints its argument on the diagnostic output file.
dumpdef	prints current names and definitions, for the named items, or for all if no arguments are given.

1. The first part of the report is devoted to a general survey of the situation in the country.	100
2. The second part of the report is devoted to a detailed analysis of the economic situation.	100
3. The third part of the report is devoted to a detailed analysis of the social situation.	100
4. The fourth part of the report is devoted to a detailed analysis of the political situation.	100
5. The fifth part of the report is devoted to a detailed analysis of the cultural situation.	100
6. The sixth part of the report is devoted to a detailed analysis of the environmental situation.	100
7. The seventh part of the report is devoted to a detailed analysis of the international situation.	100
8. The eighth part of the report is devoted to a detailed analysis of the future prospects.	100
9. The ninth part of the report is devoted to a detailed analysis of the conclusions.	100
10. The tenth part of the report is devoted to a detailed analysis of the recommendations.	100



**tracemon** with no arguments, turns on tracing for all macros (including built-ins). Otherwise, turns on tracing for named macros.

**tracemoff** turns off trace globally and for any macros specified. Macros specifically traced by *tracemon* can be untraced only by specific calls to *tracemoff*.

**SEE ALSO**

*The M4 Macro Processor* by B. W. Kernighan and D. M. Ritchie.

1. The first part of the report is a general  
description of the project and its objectives.  
2. The second part is a detailed description of the  
methodology used in the study.  
3. The third part is a description of the results  
of the study.  
4. The fourth part is a discussion of the results  
and their implications.  
5. The fifth part is a conclusion and a list of  
references.



## NAME

mail, rmail - send mail to users or read mail

## SYNOPSIS

mail [ -rpq ] [ -f file ]

mail persons

rmail persons

## DESCRIPTION

*Mail* without arguments prints a user's mail, message-by-message, in last-in, first-out order. For each message, the user is prompted with a `?`, and a line is read from the standard input to determine the disposition of the message:

<code>&lt;new-line&gt;</code>	Go on to next message.
<code>+</code>	Same as <code>&lt;new-line&gt;</code> .
<code>d</code>	Delete message and go on to next message.
<code>p</code>	Print message again.
<code>-</code>	Go back to previous message.
<code>s [ files ]</code>	Save message in the named <i>files</i> ( <i>mbox</i> is default).
<code>w [ files ]</code>	Save message, without its header, in the named <i>files</i> ( <i>mbox</i> is default).
<code>m [ persons ]</code>	Mail the message to the named <i>persons</i> (yourself is default).
<code>q</code>	Put undeleted mail back in the <i>mailfile</i> and stop.
<code>EOT (control-d)</code>	Same as <code>q</code> .
<code>x</code>	Put all mail back in the <i>mailfile</i> unchanged and stop.
<code>!command</code>	Escape to the shell to do <i>command</i> .
<code>.</code>	Print a command summary.

The optional arguments alter the printing of the mail:

- `-r` causes messages to be printed in first-in, first-out order.
- `-p` causes all mail to be printed without prompting for disposition.
- `-q` causes *mail* to terminate after interrupts. Normally an interrupt only causes the termination of the message being printed.
- `-ffile` causes *mail* to use *file* (e.g., *mbox*) instead of the default *mailfile*.

When *persons* are named, *mail* takes the standard input up to an end-of-file (or up to a line consisting of just a `.`) and adds it to each *person's* *mailfile*. The message is preceded by the sender's name and a postmark. Lines that look like postmarks in the message, (i.e., "From...") are preceded with a `>`. A *person* is usually a user name recognized by *login*(1). If a *person* being sent mail is not recognized, or if *mail* is interrupted during input, the *dead.letter* will be saved to allow editing and resending.

To denote a recipient on a remote system, prefix *person* by the system name and exclamation mark (see *uucp*(1C)). Everything after the first exclamation mark in *persons* is interpreted by the remote system. In particular, if *persons* contains additional exclamation marks, it can denote a sequence of machines through which the message is to be sent on the way to its ultimate destination. For example, specifying *a!b!cde* as a recipient's name causes the message to be sent to user *b!cde* on system *a*. System *a* will interpret that destination as a request to send the message to user *cde* on system *b*. This might be useful, for instance, if the

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000



sending system can access system a but not system b, and system a has access to system b.

The *mailfile* may be manipulated in two ways to alter the function of *mail*. The *other* permissions of the file may be read-write, read-only, or neither read nor write to allow different levels of privacy. If changed to other than the default, the file will be preserved even when empty to perpetuate the desired permissions. The file may also contain the first line:

Forward to *person*

which will cause all mail sent to the owner of the *mailfile* to be forwarded to *person*. This is especially useful to forward all of a person's mail to one machine in a multiple machine environment.

*Rmail* only permits the sending of mail; *uucp*(1C) uses *rmail* as a security precaution.

When a user logs in he is informed of the presence of mail, if any.

#### FILES

/etc/passwd	to identify sender and locate persons
/usr/mail/*	incoming mail for user *; <i>mailfile</i>
\$HOME/mbox	saved mail
\$MAIL	<i>mailfile</i>
/tmp/ma*	temporary file
/usr/mail/*.lock	lock for mail directory
dead.letter	unmailable text

#### SEE ALSO

login(1), uucp(1C), write(1).

#### BUGS

Race conditions sometimes result in a failure to remove a lock file. After an interrupt, the next message may not be printed; printing may be forced by typing a p.





## NAME

make - maintain, update, and regenerate groups of programs

## SYNOPSIS

make [-f makefile] [-p] [-i] [-k] [-s] [-r] [-n] [-b] [-e] [-m] [-t]  
[-q] [-d] [names]

## DESCRIPTION

The following is a brief description of all options and some special names:

- f *makefile* Description file name. *Makefile* is assumed to be the name of a description file. A file name of - denotes the standard input. The contents of *makefile* override the built-in rules if they are present.
- p Print out the complete set of macro definitions and target descriptions.
- i Ignore error codes returned by invoked commands. This mode is entered if the fake target name .IGNORE appears in the description file.
- k Abandon work on the current entry, but continue on other branches that do not depend on that entry.
- s Silent mode. Do not print command lines before executing. This mode is also entered if the fake target name .SILENT appears in the description file.
- r Do not use the built-in rules.
- n No execute mode. Print commands, but do not execute them. Even lines beginning with an @ are printed.
- b Compatibility mode for old makefiles.
- e Environment variables override assignments within makefiles.
- m Print a memory map showing text, data, and stack. This option is a no-operation on systems without the *getu* system call.
- t Touch the target files (causing them to be up-to-date) rather than issue the usual commands.
- d Debug mode. Print out detailed information on files and times examined.
- q Question. The *make* command returns a zero or non-zero status code depending on whether the target file is or is not up-to-date.
- .DEFAULT If a file must be made but there are no explicit commands or relevant built-in rules, the commands associated with the name .DEFAULT are used if it exists.
- .PRECIOUS Dependents of this target will not be removed when quit or interrupt are hit.
- .SILENT Same effect as the -s option.

THE FIRST PART OF THE HISTORY OF THE

REIGN OF THE GREAT KING OF THE

THE SECOND PART OF THE HISTORY OF THE

THE THIRD PART OF THE HISTORY OF THE

THE FOURTH PART OF THE HISTORY OF THE

THE FIFTH PART OF THE HISTORY OF THE

THE SIXTH PART OF THE HISTORY OF THE

THE SEVENTH PART OF THE HISTORY OF THE

THE EIGHTH PART OF THE HISTORY OF THE

THE NINTH PART OF THE HISTORY OF THE

THE TENTH PART OF THE HISTORY OF THE

THE ELEVENTH PART OF THE HISTORY OF THE

THE TWELFTH PART OF THE HISTORY OF THE

THE THIRTEENTH PART OF THE HISTORY OF THE

THE FOURTEENTH PART OF THE HISTORY OF THE

THE FIFTEENTH PART OF THE HISTORY OF THE

THE SIXTEENTH PART OF THE HISTORY OF THE

THE SEVENTEENTH PART OF THE HISTORY OF THE

THE EIGHTEENTH PART OF THE HISTORY OF THE

THE NINETEENTH PART OF THE HISTORY OF THE



**.IGNORE** Same effect as the **-i** option.

*Make* executes commands in *makefile* to update one or more target *names*. *Name* is typically a program. If no **-f** option is present, *makefile*, *Makefile*, *s.makefile*, and *s.Makefile* are tried in order. If *makefile* is **-**, the standard input is taken. More than one **-f** *makefile* argument pair may appear.

*Make* updates a target only if it depends on files that are newer than the target. All prerequisite files of a target are added recursively to the list of targets. Missing files are deemed to be out of date.

*Makefile* contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated, non-null list of targets, then a **:**, then a (possibly null) list of prerequisite files or dependencies. Text following a **:** and all following lines that begin with a tab are shell commands to be executed to update the target. The first line that does not begin with a tab or **#** begins a new dependency or macro definition. Shell commands may be continued across lines with the **<backslash><new-line>** sequence. Sharp (**#**) and new-line surround comments.

The following *makefile* says that *pgm* depends on two files *a.o* and *b.o*, and that they in turn depend on their corresponding source files (*a.c* and *b.c*) and a common file *incl.h*:

```
pgm: a.o b.o
    cc a.o b.o -o pgm
a.o: incl.h a.c
    cc -c a.c
b.o: incl.h b.c
    cc -c b.c
```

Command lines are executed one at a time, each by its own shell. A line is printed when it is executed unless the **-s** option is present, or the entry **.SILENT:** is in *makefile*, or unless the first character of the command is **@**. The **-n** option specifies printing without execution; however, if the command line has the string **\$(MAKE)** in it, the line is always executed (see discussion of the **MAKEFLAGS** macro under *Environment*). The **-t** (touch) option updates the modified date of a file without executing any commands.

Commands returning non-zero status normally terminate *make*. If the **-i** option is present, or the entry **.IGNORE:** appears in *makefile*, or if the line specifying the command begins with **<tab><hyphen>**, the error is ignored. If the **-k** option is present, work is abandoned on the current entry, but continues on other branches that do not depend on that entry.

The **-b** option allows old *makefiles* (those written for the old version of *make*) to run without errors. The difference between the old version of *make* and this version is that this version requires all dependency lines to have a (possibly null) command associated with them. The previous version of *make* assumed if no command was specified explicitly that the command was null.

Interrupt and quit cause the target to be deleted unless the target depends on the special name **.PRECIOUS**.

The first of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.

The second of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.

The third of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.

The fourth of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.

The fifth of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.

The sixth of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.

The seventh of these is the fact that the system is not a simple one. It is a complex system, and the complexity is not only in the number of components, but also in the way they are connected. The system is a complex system, and the complexity is not only in the number of components, but also in the way they are connected.



## Environment

The environment is read by *make*. All variables are assumed to be macro definitions and processed as such. The environment variables are processed before any makefile and after the internal rules; thus, macro assignments in a makefile override environment variables. The *-e* option causes the environment to override the macro assignments in a makefile.

The **MAKEFLAGS** environment variable is processed by *make* as containing any legal input option (except *-f*, *-p*, and *-d*) defined for the command line. Further, upon invocation, *make* "invents" the variable if it is not in the environment, puts the current options into it, and passes it on to invocations of commands. Thus, **MAKEFLAGS** always contains the current input options. This proves very useful for "super-makes". In fact, as noted above, when the *-n* option is used, the command **\$(MAKE)** is executed anyway; hence, one can perform a *make -n* recursively on a whole software system to see what would have been executed. This is because the *-n* is put in **MAKEFLAGS** and passed to further invocations of **\$(MAKE)**. This is one way of debugging all of the makefiles for a software project without actually doing anything.

## Macros

Entries of the form *string1 = string2* are macro definitions. Subsequent appearances of **\$(string1[:subst1=[subst2]])** are replaced by *string2*. The parentheses are optional if a single character macro name is used and there is no substitute sequence. The optional *:subst1=subst2* is a substitute sequence. If it is specified, all non-overlapping occurrences of *subst1* in the named macro are replaced by *subst2*. Strings (for the purposes of this type of substitution) are delimited by blanks, tabs, new-line characters, and beginnings of lines. An example of the use of the substitute sequence is shown under *Libraries*.

## Internal Macros

There are five internally maintained macros which are useful for writing rules for building targets.

- \$\*** The macro **\$\*** stands for the file name part of the current dependent with the suffix deleted. It is evaluated only for inference rules.
- \$@** The **\$@** macro stands for the full target name of the current target. It is evaluated only for explicitly named dependencies.
- \$<** The **\$<** macro is only evaluated for inference rules or the **.DEFAULT** rule. It is the module which is out of date with respect to the target (i.e., the "manufactured" dependent file name). Thus, in the **.co** rule, the **\$<** macro would evaluate to the **.c** file. An example for making optimized **.o** files from **.c** files is:

```
.c.o:
    cc -c -O $*.c
```

or:

```
.c.o:
    cc -c -O $<
```

- \$?** The **\$?** macro is evaluated when explicit rules from the makefile are evaluated. It is the list of prerequisites that are out of date with respect to the target; essentially, those modules which must be





rebuilt.

**\$%** The **\$%** macro is only evaluated when the target is an archive library member of the form **lib(file.o)**. In this case, **\$@** evaluates to **lib** and **\$%** evaluates to the library member, **file.o**.

Four of the five macros can have alternative forms. When an upper case **D** or **F** is appended to any of the four macros the meaning is changed to "directory part" for **D** and "file part" for **F**. Thus, **\$(@D)** refers to the directory part of the string **\$@**. If there is no directory part, The only macro excluded from this alternative form is **\$?**. The reasons for this are debatable.

### Suffixes

Certain names (for instance, those ending with **.o**) have inferable prerequisites such as **.c**, **.s**, etc. If no update commands for such a file appear in *makefile*, and if an inferable prerequisite exists, that prerequisite is compiled to make the target. In this case, *make* has inference rules which allow building files from other files by examining the suffixes and determining an appropriate inference rule to use. The current default inference rules are:

```
.c .c~ .sh .sh~ .c.o .c~.o .c~.c .s.o .s~.o .y.o .y~.o .l.o .l~.o
.y.c .y~.c .l.c .c.a .c~.a .s~.a .h~.h
```

The internal rules for *make* are contained in the source file **rules.c** for the *make* program. These rules can be locally modified. To print out the rules compiled into the *make* on any machine in a form suitable for recompilation, the following command is used:

```
make -fp - 2>/dev/null </dev/null
```

The only peculiarity in this output is the **(null)** string which *printf*(3S) prints when handed a null string.

A tilde in the above rules refers to an SCCS file (see *sccsfile*(5)). Thus, the rule **.c~.o** would transform an SCCS C source file into an object file (**.o**). Because the **s** of the SCCS files is a prefix it is incompatible with *make*'s suffix point-of-view. Hence, the tilde is a way of changing any file reference into an SCCS file reference.

A rule with only one suffix (i.e. **.c**) is the definition of how to build **x** from **x.c**. In effect, the other suffix is null. This is useful for building targets from only one source file (e.g., shell procedures, simple C programs).

Additional suffixes are given as the dependency list for **.SUFFIXES**. Order is significant; the first possible name for which both a file and a rule exist is inferred as a prerequisite.

The default list is:

```
.SUFFIXES: .o .c .y .l .s
```

Here again, the above command for printing the internal rules will display the list of suffixes implemented on the current machine. Multiple suffix lists accumulate; **.SUFFIXES:** with no dependencies clears the list of suffixes.

### Inference Rules

The first example can be done more briefly:





```

pgm: a.o b.o
      cc a.o b.o -o pgm
a.o b.o: incl.h

```

This is because *make* has a set of internal rules for building files. The user may add rules to this list by simply putting them in the *makefile*.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, **CFLAGS**, **LFLAGS**, and **YFLAGS** are used for compiler options to *cc*(1), *lex*(1), and *yacc*(1) respectively. Again, the previous method for examining the current rules is recommended.

The inference of prerequisites can be controlled. The rule to create a file with suffix **.o** from a file with suffix **.c** is specified as an entry with **.c.o**: as the target and no dependents. Shell commands associated with the target define the rule for making a **.o** file from a **.c** file. Any target that has no slashes in it and starts with a dot is identified as a rule and not a true target.

### Libraries

If a target or dependency name contains parenthesis, it is assumed to be an archive library, the string within parenthesis referring to a member within the library. Thus **lib(file.o)** and **\$(LIB)(file.o)** both refer to an archive library which contains **file.o**. (This assumes the **LIB** macro has been previously defined.) The expression **\$(LIB)(file1.o file2.o)** is not legal. Rules pertaining to archive libraries have the form **XX.a** where the **XX** is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires the **XX** to be different from the suffix of the archive member. Thus, one cannot have **lib(file.o)** depend upon **file.o** explicitly. The most common use of the archive interface follows. Here, we assume the source files are all C type source:

```

lib:   lib(file1.o) lib(file2.o) lib(file3.o)
      @echo lib is now up to date

.c.a:
      $(CC) -c $(CFLAGS) $<
      ar rv $@ $*.o
      rm -f $*.o

```

In fact, the **.c.a** rule listed above is built into *make* and is unnecessary in this example. A more interesting, but more limited example of an archive library maintenance construction follows:

```

lib:   lib(file1.o) lib(file2.o) lib(file3.o)
      $(CC) -c $(CFLAGS) $(?:.o=.c)
      ar rv lib $?
      rm $? @echo lib is now up to date

.c.a:;

```

Here the substitution mode of the macro expansions is used. The **\$(?)** list is defined to be the set of object file names (inside **lib**) whose C source files are out of date. The substitution mode translates the **.o** to **.c**. (Unfortunately, one cannot as yet transform to **.c~**; however, this may become possible in the future.) Note also, the disabling of the **.c.a**: rule, which would have created each object file, one by one. This particular

Dear Sir,  
I have the honor to acknowledge the receipt of your letter of the 10th inst. in relation to the above matter.  
The same has been forwarded to the proper authorities for their consideration.  
I am, Sir, very respectfully,  
Yours,  
[Signature]

I am, Sir, very respectfully,  
Yours,  
[Signature]

I am, Sir, very respectfully,  
Yours,  
[Signature]

I am, Sir, very respectfully,  
Yours,  
[Signature]



construct speeds up archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

**FILES**

[Mm]akefile  
s.[Mm]akefile

**SEE ALSO**

sh(1), mk(8).

*Make-A Program for Maintaining Computer Programs* by S. I. Feldman.  
*An Augmented Version of Make* by E. G. Bradford.

**BUGS**

Some commands return non-zero status inappropriately; use `-i` to overcome the difficulty. Commands that are directly executed by the shell, notably `cd(1)`, are ineffectual across new-lines in *make*. The syntax `lib(file1.o file2.o file3.o)` is illegal. You cannot build `lib(file.o)` from `file.o`. The macro `$(a:.o=.c~)` doesn't work.

The first of these is the fact that the  
the second is the fact that the  
the third is the fact that the

the fourth is the fact that the  
the fifth is the fact that the

the sixth is the fact that the  
the seventh is the fact that the

the eighth is the fact that the  
the ninth is the fact that the  
the tenth is the fact that the



## NAME

move - copy directories

## SYNOPSIS

move *fromdir todir*

## DESCRIPTION

Move copies the directory *fromdir* with all its subdirectories to the directory *todir*.

The directory *todir* will be created if necessary.

## SEE ALSO

cpio(1)

1. The first part of the report discusses the general situation of the country and the progress of the work during the year. It also mentions the results of the various committees and the work of the different departments.

2. The second part of the report deals with the financial situation of the country and the progress of the work during the year. It also mentions the results of the various committees and the work of the different departments.

3. The third part of the report discusses the general situation of the country and the progress of the work during the year. It also mentions the results of the various committees and the work of the different departments.

4. The fourth part of the report deals with the financial situation of the country and the progress of the work during the year. It also mentions the results of the various committees and the work of the different departments.

5. The fifth part of the report discusses the general situation of the country and the progress of the work during the year. It also mentions the results of the various committees and the work of the different departments.



**NAME**

**newconf** - generate configuration file and reconfigure **MUNIX**

**SYNOPSIS**

**/etc/newconf** [ configuration file ]

**DESCRIPTION**

*Newconf* makes a new **MUNIX** kernel from an already existing or from an interactively created configuration file.

If no configuration file is specified *newconf* asks for

- the device drivers to be included  
(tty driver is automatically included)
- assignation of DMA extension registers to DMA devices
- the type and unit of the root and swap device
- the origin (block number) and size of the swap area
- some other system parameters

*Newconf* creates the configuration file **/usr/sys/conf.h** as include file to the configuration table **/usr/sys/c.c** and the interrupt vector table **/usr/sys/l.s**.

If a configuration file is specified this file is copied to **/usr/sys/conf.h**.

Reconfiguration of **MUNIX** is automatically done by compiling **/usr/sys/c.c**, assembling **/usr/sys/l.s** and linking **/usr/sys/c.o**, **/usr/sys/l.o**, the kernel library **/usr/sys/lib1** and the driver libraries **/usr/sys/lib2** and **/usr/sys/lib3** to a new **MUNIX** kernel named **/nunix**.

For later use you should save the interactively created configuration file **/usr/sys/conf.h** by renaming it.

Several drivers exist in two versions (for 18-bit and 22-bit controllers). These versions you can find in **/usr/sys/libchoice**. The appropriate version of each of those drivers (for your configuration) should be in **/usr/sys/lib3**. If the driver versions in **/usr/sys/lib3** do not match your needs you have to exchange the wrong versions.

**FILES**

**/usr/sys/conf.h**  
**/usr/sys/c.\***  
**/usr/sys/l.\***  
**/usr/sys/lib1**  
**/usr/sys/lib2**  
**/usr/sys/lib3**  
**/usr/sys/libchoice**  
**/usr/sys/confinfo**  
**/nunix**

**SEE ALSO**

**whatconf (1m)**  
**confinfo (7)**

**BUGS**





**NAME**

**newgrp** - log in to a new group

**SYNOPSIS**

**newgrp** [ group ]

**DESCRIPTION**

*Newgrp* changes the group identification of its caller, analogously to *login*(1). The same person remains logged in, and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new group ID.

*Newgrp* without an argument changes the group identification to the group in the password file; in effect it changes the group identification back to the caller's original group.

A password is demanded if the group has a password and the user himself does not, or if the group has a password and the user is not listed in */etc/group* as being a member of that group.

When most users log in, they are members of the group named **other**.

**FILES**

*/etc/group*  
*/etc/passwd*

**SEE ALSO**

*login*(1), *group*(5).

**BUGS**

There is no convenient way to enter a password into */etc/group*. Use of group passwords is not encouraged, because, by their very nature, they encourage poor security practices. Group passwords may disappear in the future.

THE FOLLOWING IS A SUMMARY OF THE  
RESULTS OF THE INVESTIGATION  
CONDUCTED BY THE BUREAU OF  
THE ARMY OF THE UNITED STATES  
ON THE SUBJECT OF THE  
ALLEGED VIOLATIONS OF THE  
MILITARY LAWS AND REGULATIONS  
BY THE OFFICERS AND MEN  
OF THE ARMY OF THE UNITED STATES  
WHO WERE ENGAGED IN THE  
OPERATIONS OF THE ARMY OF THE  
UNITED STATES IN THE  
MEXICAN WAR.

THE RESULTS OF THE INVESTIGATION  
CONDUCTED BY THE BUREAU OF  
THE ARMY OF THE UNITED STATES  
ON THE SUBJECT OF THE  
ALLEGED VIOLATIONS OF THE  
MILITARY LAWS AND REGULATIONS  
BY THE OFFICERS AND MEN  
OF THE ARMY OF THE UNITED STATES  
WHO WERE ENGAGED IN THE  
OPERATIONS OF THE ARMY OF THE  
UNITED STATES IN THE  
MEXICAN WAR.



## NAME

nl - line numbering filter

## SYNOPSIS

nl [-h`type`] [-b`type`] [-f`type`] [-v`start#`] [-i`incr`] [-p] [-l`num`] [-s`sep`]  
[-w`width`] [-n`format`] `file`

## DESCRIPTION

*Nl* reads lines from the named *file* or the standard input if no *file* is named and reproduces the lines on the standard output. Lines are numbered on the left in accordance with the command options in effect.

*Nl* views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page. A logical page consists of a header, a body, and a footer section. Empty sections are valid. Different line numbering options are independently available for header, body, and footer (e.g. no numbering of header and footer lines while numbering blank lines only in the body).

The start of logical page sections are signaled by input lines containing nothing but the following character(s):

*Line contents*    *Start of*

\\:\\:\\:            header

\\:\\:                body

\\:                   footer

Unless signaled otherwise, *nl* assumes the text being read is in a single logical page body.

Command options may appear in any order and may be intermingled with an optional file name. Only one file may be named. The options are:

- b`type`    Specifies which logical page body lines are to be numbered. Recognized *types* and their meaning are: *a*, number all lines; *t*, number lines with printable text only; *n*, no line numbering; *pstring*, number only lines that contain the regular expression specified in *string*. Default *type* for logical page body is *t* (text lines numbered).
- h`type`    Same as -b`type` except for header. Default *type* for logical page header is *n* (no lines numbered).
- f`type`    Same as -b`type` except for footer. Default for logical page footer is *n* (no lines numbered).
- p            Do not restart numbering at logical page delimiters.
- v`start#`   *Start#* is the initial value used to number logical page lines. Default is 1.
- i`incr`    *incr* is the increment value used to number logical page lines. Default is 1.
- s`sep`      *Sep* is the character(s) used in separating the line number and the corresponding text line. Default *sep* is a tab.
- w`width`   *Width* is the number of characters to be used for the line number. Default *width* is 6.





**-nformat**

*Format* is the line numbering format. Recognized values are: *ln*, left justified, leading zeroes suppressed; *rn*, right justified, leading zeroes suppressed; *rz*, right justified, leading zeroes kept. Default *format* is *rn* (right justified).

**-inum** *Num* is the number of blank lines to be considered as one. For example, **-12** results in only the second adjacent blank being numbered (if the appropriate **-ha**, **-ba**, and/or **-fa** option is set). Default is 1.

SEE ALSO  
pr(1).

The first part of the report deals with the general situation of the country. It is a very interesting and informative study of the country's development. The second part of the report deals with the specific details of the country's development. It is a very detailed and thorough study of the country's development. The third part of the report deals with the future of the country. It is a very optimistic and hopeful study of the country's future.

103





## NAME

**paste** - merge same lines of several files or subsequent lines of one file

## SYNOPSIS

```
paste file1 file2 ...
paste -dlist file1 file2 ...
paste -s [-dlist] file1 file2 ...
```

## DESCRIPTION

In the first two forms, *paste* concatenates corresponding lines of the given input files *file1*, *file2*, etc. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). If you will, it is the counterpart of *cat*(1) which concatenates vertically, i.e., one file after the other. In the last form above, *paste* subsumes the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the *tab* character, or with characters from an optionally specified *list*. Output is to the standard output, so it can be used as the start of a pipe, or as a filter, if *-* is used in place of a file name.

The meanings of the options are:

- d** Without this option, the new-line characters of each but the last file (or last line in case of the **-s** option) are replaced by a *tab* character. This option allows replacing the *tab* character by one or more alternate characters (see below).
- list** One or more characters immediately following **-d** replace the default *tab* as the line concatenation character. The list is used circularly, i. e. when exhausted, it is reused. In parallel merging (i. e. no **-s** option), the lines from the last file are always terminated with a new-line character, not from the *list*. The list may contain the special escape sequences: **\n** (new-line), **\t** (tab), **\\** (backslash), and **\0** (empty string, not a null character). Quoting may be necessary, if characters have special meaning to the shell (e.g. to get one backslash, use **-d"\\\\"**).
- s** Merge subsequent lines rather than one from each input file. Use *tab* for concatenation, unless a *list* is specified with **-d** option. Regardless of the *list*, the very last character of the file is forced to be a new-line.
- May be used in place of any file name, to read a line from the standard input. (There is no prompting).

## EXAMPLES

```
ls | paste -d" " -      list directory in one column
ls | paste - - - -      list directory in four columns
paste -s -d"\ t\ n" file  combine pairs of lines into lines
```

## SEE ALSO

*grep*(1), *cut*(1),  
*pr*(1): *pr -t -m..* works similarly, but creates extra blanks, tabs and new-lines for a nice page layout.





DIAGNOSTICS

*line too long*

*too many files*

Output lines are restricted to 511 characters.

Except for **-s** option, no more than 12 input files may be specified.





## NAME

ped - "screen oriented editor"

## SYNOPSIS

ped <file>

## DESCRIPTION

Ped is an interactive, screen oriented editor. <file> is the name of the file to be edited. Ped is initially in REPLACE mode: move the cursor around the screen, then type away. ENTER i, ENTER l etc. switch to the following modes:

REPLACE  
INSERT  
LINE  
FIND  
USE  
WINDOW POSIT  
SAVE  
END EDITING  
AREA  
PARAGRAPH  
TAG  
CHANGE KEYS  
COMMAND

Cursor position with the four arrow keys: , ↑ , <- , and -> . The HOME key 'amplifies' the following cursor key, e.g. HOME -> moves the cursor to the right margin of the window. If the cursor is already at the right margin, HOME -> moves the window half a page right. Similarly, HOME ↑ moves the cursor to the top margin, or moves up half a page. HOME HOME moves the cursor to the limits of the text, in this case to the last line.

Delete a character: hitting DEL CHAR deletes the character at the cursor position; DELETE or BACK SPACE deletes the character left of the cursor position.

Hitting ENTER switches into the COMMAND mode, and gives a 'menu' list of the various commands.

LINE operations: delete (d), insert (i), split (s), remove the rest of (r) a line.

FIND: ENTER f find-string finds the next occurrence of 'find-string' in the file.

ENTER f looks for the same 'find-string' again. ENTER f ↑ looks up instead of down.





USE: ENTER u <file name> RETURN switches to another file.

WINDOW POSITION: ENTER w line number RETURN or or ↑ moves to a given line number in the file. ENTER w +20 goes down 20 lines, ENTER w -20 ↑ goes up 20 lines, ENTER w +20 -> goes right 20 columns.

SAVE, END EDITING asks if you really want to change the file. Answer y to save, or n to continue editing. END EDITING closes the file and goes back to Unix.

In AREA mode, first mark a range of lines by:

- m -- mark the top line,
- move the cursor,
- m -- mark the bottom line.

Then you can delete (d), insert blank lines (i), copy (c) the marked lines. (Copy copies just below the current cursor position).

PARAGRAPH is like AREA, for rectangles instead of line ranges. First:

- M -- mark the corner of a rectangle,
- move the cursor,
- M -- mark the opposite corner of the rectangle.

Then delete (d), horizontal insert (h), vertical insert (v), replace (r) with the upper left corner of the rectangle at the current cursor position.

CALLING: If you call ped then ped looks for a description file in the directory /usr/lib/red/PE<tty number>.<username>. The contents of such a file is the file name of the last file you edited and the position in that file. If such a file exists for your terminal number and user name, then ped takes the last file you edited and positions where you left it. If you call ped -f search or you type ped -w line then ped positions in the last file edited to the string search or the line line.

TAG: If you call ped -t search or you type ENTER t search RETURN or or ↑ then ped looks in the file tags in the current directory for the entry search. (See also ctags (1)). It opens the corresponding file and positions to the line where procedure or function search is declared. This works for C and PASCAL files.

CHANGE KEYS: to expand a single letter to a long word, like

Dear Sir,

I have the pleasure to acknowledge the receipt of your letter of the 15th inst. in relation to the above matter.

It is noted that you are desirous of obtaining further information regarding the same.

Enclosed herewith are the documents which you have requested.

I am sure that this information will be of assistance to you.

Very truly yours,

John Doe

Enclosure

Very truly yours,

John Doe

Enclosure

Very truly yours,

John Doe

Enclosure



'M' to 'Mississippi', type:

ENTER ENTER <delimiter> <Mississippi> <delimiter> <M>.

Then hitting 'M' is the same as typing in 'Mississippi'.

The expanded 'word' can also contain editor commands, as in:

/ENTER f ENTER r replace/M

The ENTER x command displays all expand keys, in the format  
<expand key>: <sequence for this key>.

The ENTER y command escapes to the unix shell. EOT  
(control-z) returns to Ped.

## KEYBOARD

The CURSOR keys and some other special keys are different from keyboard to keyboard. Therefore the /etc/termcap data base contains a translation for these keys to an internal representation. See also termcap (3) and termcap (5) . The new termcap entries all start with 'y'.

y1 => ENTER  
y2 => DELETE  
y3 => BACKSPACE  
y4 => TAB  
y5 => DEL CHAR  
y6 => RETURN  
y7 => BACKTAB  
y8 => REFRESH  
y9 => KILL LINE  
ya => + PAGE  
yb => - PAGE  
yc => + LINE  
yd => - LINE  
ye => INSERT LINE  
yf => DELETE LINE  
yg => CURSOR UP  
yh => CURSOR DOWN  
yi => CURSOR RIGHT  
yj => CURSOR LEFT  
yk => CURSOR HOME

If your keyboard has some function keys, then you can install additional functions:

+/- PAGE (LINE) moves the text in the window up/down one (half a) page. INSERT, DELETE LINE inserts or deletes a line without changing the current mode.

## FILES

/tmp/text<pid>  
/etc/termcap  
/usr/lib/red/PE<ttynumber>.<username>

The first thing I noticed when I stepped out of the car was the cold. It was a sharp contrast to the warm blanket I had been under. I looked up at the sky, which was a pale, overcast grey. The air was still, and the silence was broken only by the distant hum of a car or the occasional cough of a person. I felt a sense of isolation, as if I were the only one in the world.

As I walked, I noticed the texture of the ground beneath my feet. It was a mix of dirt and gravel, and the sound of my shoes hitting it was a rhythmic accompaniment to my thoughts.

I had heard that the weather was bad, but I didn't realize how much it would affect my mood. The grey sky and the cold air seemed to seep into my bones, making me feel like a stranger in a strange land. I tried to shake the feeling off, but it was there, lurking in the corners of my mind.

The first thing I noticed when I stepped out of the car was the cold. It was a sharp contrast to the warm blanket I had been under. I looked up at the sky, which was a pale, overcast grey. The air was still, and the silence was broken only by the distant hum of a car or the occasional cough of a person. I felt a sense of isolation, as if I were the only one in the world.

I had heard that the weather was bad, but I didn't realize how much it would affect my mood. The grey sky and the cold air seemed to seep into my bones, making me feel like a stranger in a strange land. I tried to shake the feeling off, but it was there, lurking in the corners of my mind.

The first thing I noticed when I stepped out of the car was the cold. It was a sharp contrast to the warm blanket I had been under. I looked up at the sky, which was a pale, overcast grey. The air was still, and the silence was broken only by the distant hum of a car or the occasional cough of a person. I felt a sense of isolation, as if I were the only one in the world.

I had heard that the weather was bad, but I didn't realize how much it would affect my mood. The grey sky and the cold air seemed to seep into my bones, making me feel like a stranger in a strange land. I tried to shake the feeling off, but it was there, lurking in the corners of my mind.



**AUTHOR**

Arnulf P.A. Wiedemann  
ZT ZTI INF 212  
Siemens AG  
Otto-Hahn-Ring 6  
D 8000 Muenchen 83

**BUGS**

AREA is not well tested.





## NAME

**pr** - print files

## SYNOPSIS

**pr** [ options ] [ files ]

## DESCRIPTION

**Pr** prints the named files on the standard output. If *file* is **-**, or if no files are specified, the standard input is assumed. By default, the listing is separated into pages, each headed by the page number, a date and time, and the name of the file.

By default, columns are of equal width, separated by at least one space; lines which do not fit are truncated. If the **-s** option is used, lines are not truncated and columns are separated by the separation character.

If the standard output is associated with a terminal, error messages are withheld until **pr** has completed printing.

Options may appear singly or be combined in any order. Their meanings are:

- +k** Begin printing with page *k* (default is 1).
- k** Produce *k*-column output (default is 1). The options **-e** and **-i** are assumed for multi-column output.
- a** Print multi-column output across the page.
- m** Merge and print all files simultaneously, one per column (overrides the **-k**, and **-a** options).
- d** Double-space the output.
- eck** Expand *input* tabs to character positions *k*+1, 2•*k*+1, 3•*k*+1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. Tab characters in the input are expanded into the appropriate number of spaces. If *c* (any non-digit character) is given, it is treated as the input tab character (default for *c* is the tab character).
- ick** In *output*, replace white space wherever possible by inserting tabs to character positions *k*+1, 2•*k*+1, 3•*k*+1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any non-digit character) is given, it is treated as the output tab character (default for *c* is the tab character).
- nck** Provide *k*-digit line numbering (default for *k* is 5). The number occupies the first *k*+1 character positions of each column of normal output or each line of **-m** output. If *c* (any non-digit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a tab).
- wk** Set the width of a line to *k* character positions (default is 72 for equal-width multi-column output, no limit otherwise).
- ok** Offset each line by *k* character positions (default is 0). The number of character positions per line is the sum of the width and offset.

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...



- lk** Set the length of a page to *k* lines (default is 66).
- h** Use the next argument as the header to be printed instead of the file name.
- p** Pause before beginning each page if the output is directed to a terminal (*pr* will ring the bell at the terminal and wait for a carriage return).
- f** Use form-feed character for new pages (default is to use a sequence of line-feeds). Pause before beginning the first page if the standard output is associated with a terminal.
- r** Print no diagnostic reports on failure to open files.
- t** Print neither the five-line identifying header nor the five-line trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.
- sc** Separate columns by the single character *c* instead of by the appropriate number of spaces (default for *c* is a tab).

**EXAMPLES**

Print *file1* and *file2* as a double-spaced, three-column listing headed by "file list":

```
pr -3dh "file list" file1 file2
```

Write *file1* on *file2*, expanding tabs to columns 10, 19, 28, 37, ...:

```
pr -e9 -t <file1 >file2
```

**FILES**

/dev/tty\* to suspend messages

**SEE ALSO**

cat(1).

On the 1st of January 1900, the first of the year, the weather was very cold and the wind was strong from the north.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.

The day was very cold and the wind was strong from the north. The sun was not out and the sky was very dark.



**NAME**

print - print files on the line printer

**SYNOPSIS**

print file ...

**DESCRIPTION**

*Print* produces a printed listing of one or more *files* with line numbers on the line printer. The output is preceeded by a banner page with the user's login name. A printed file is separated into pages headed by the date, the filename and the page number.

**FILES**

/dev/tty? to suspend messages.

**SEE ALSO**

lpr(1), pr(1)





**NAME**

pwck, grpck — password/group file checkers

**SYNOPSIS**

pwck [file]  
grpck [file]

**DESCRIPTION**

*Pwck* scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program name exist. The criteria for determining a valid login name are taken from *Setting Up UNIX*. The default password file is */etc/passwd*.

*Grpck* verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is */etc/group*.

**FILES**

*/etc/group*  
*/etc/passwd*

**SEE ALSO**

group(5), passwd(5).  
*Setting Up UNIX*.

**DIAGNOSTICS**

Group entries in */etc/group* with no login names are flagged.

1940

1940

1940

1940

1940

1940

1940

1940

1940

1940



## NAME

ratfor - rational Fortran dialect

## SYNOPSIS

ratfor [ options ] [ files ]

## DESCRIPTION

*Ratfor* converts a rational dialect of Fortran into ordinary irrational Fortran. *Ratfor* provides control flow constructs essentially identical to those in C:

statement grouping:

{ statement; statement; statement }

decision-making:

if (condition) statement [ else statement ]

switch (integer value) {

case integer: statement

...

[ default: ] statement

}

loops:

while (condition) statement

for (expression; condition; expression) statement

do limits statement

repeat statement [ until (condition) ]

break

next

and some syntactic sugar to make programs easier to read and write:

free form input:

multiple statements/line; automatic continuation

comments:

# this is a comment.

translation of relationals:

>, >=, etc., become .GT., .GE., etc.

return expression to caller from function:

return (expression)

define:

define name replacement

include:

include file

The option **-h** causes quoted strings to be turned into 27H constructs. The **-C** option copies comments to the output and attempts to format it neatly. Normally, continuation lines are marked with a **&** in column 1; the option **-6x** makes the continuation character **x** and places it in column 6.

*Ratfor* is best used with *f77(1)*.

## SEE ALSO

*efl(1)*, *f77(1)*.

B. W. Kernighan and P. J. Plauger, *Software Tools*, Addison-Wesley, 1976.

Journal of the [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



## NAME

*reform* — reformat text file

## SYNOPSIS

*reform* [tabspec1 [tabspec2]] [+bn] [+en] [+f] [+in] [+mn] [+pn] [+s]  
[+tn]

## DESCRIPTION

*Reform* reads each line of the standard input file, reformats it, and then writes it to the standard output. Various combinations of reformatting operations can be selected, of which the most common involve rearrangement of tab characters. It is often used to trim trailing blanks, truncate lines to a specified length, or prepend blanks to lines.

*Reform* first scans its arguments, which may be given in any order. It then processes its input file, performing the following actions upon each line, in the order given:

- A line is read from the standard input.
- If *+s* is given, all characters up to the first tab are stripped off and saved for later addition to the end of the line. Presumably, these characters comprise an "SCCS SID" produced by *get(1)*.
- The line is expanded into a tabless form, by replacing tabs with blanks according to the *input* tab specification *tabspec1*.
- If *+pn* is given, *n* blanks are prepended to the line.
- If *+tn* is given, the line is truncated to a length of *n* characters.
- All trailing blanks are now removed.
- If *+en* is included, the line is extended out with blanks to the length of *n* characters.
- If *+s* is given, the previously-saved "SCCS SID" is added to the end of the line.
- If *+bn* is given, the *n* characters at the beginning of the line are converted to blanks, if and only if all of them are either digits or blanks.
- If *+mn* is included, the line is moved left, i.e., *n* characters are removed from the beginning of the line.
- The line is now contracted by replacing some blanks with tab characters according to the list of tabs indicated by the *output* tab specification *tabspec2*, and is written to the standard output file. Option *+i* controls the method of contraction (see below).

The various arguments accepted by *reform* are as follows:

*tabspec1*

describes the tab stops assumed for the input file. This tab specification may take on any of the forms described in *tabs(1)*. In addition, the operand *—* indicates that the tab specification is to be found in the first line read from the standard input. If no legal tab specification is found there, *-8* is assumed. If *tabspec1* is omitted entirely, *—* is assumed.

*tabspec2*

describes the tabs assumed for the output file. It is

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...



interpreted in the same way as *tabspec1*, except that omission of *tabspec2* causes the value of *tabspec1* to be used for *tabspec2*.

The remaining arguments are all optional and may be used in any combination, although only a few combinations make much sense. Specifying an argument causes an action to be performed, as opposed to the usual default of not performing the action. Some options include numeric values, which also have default values. Option actions are applied to each line in the order described above. Any line length mentioned applies to the length of a line just before the execution of the option described, and the terminating new-line is never counted in the line length.

- +bn** causes the first *n* characters of a line to be converted to blanks, if and only if those characters include only blanks and digits. If *n* is omitted, the default value is 6, which is useful in deleting sequence numbers from COBOL programs.
- +en** causes each line shorter than *n* characters to be extended out with blanks to that length. Omitting *n* implies a default value of 72. This option is useful for those rare cases in which sequence numbers need to be added to an existing unnumbered file. The use of **\$** in editor regular expressions is more convenient if all lines have equal length, so that the user can issue editor commands such as:  
s/\$00001000/
- +f** causes a format line to be written to the standard output, preceding any other lines written. See *fspec(5)* for details regarding format specifications. The format line is taken from *tabspec2*, i.e., the line normally appears as follows:  
<t-tabspec2 d:>

If *tabspec2* is of the form —*file-name* (i.e., an indirect reference to a tab specification in the first line of the named file), then that tab specification line is written to the standard output.

- +in** controls the technique used to compress interior blanks into tabs. Unless this option is specified, any sequence of 1 or more blanks may be converted to a single tab character if that sequence occurs just before a tab stop. This causes no problems for blanks that occur before the first nonblank character in a line, and it is always possible to convert the result back to an equivalent tabless form. However, occasionally an interior blank (one occurring after the first nonblank) is converted to a tab when this is not intended. For instance, this might occur in any program written in a language utilizing blanks as delimiters. Any single blank might be converted to a tab if it occurred just before a tab stop. Insertion or deletion of characters preceding such a tab may cause it to be interpreted in an unexpected way at a later time. If the **+i** option is used, no string of blanks may be converted to a tab unless there are *n* or more contiguous blanks. The default value is 2. Note that leading blanks are always converted to tabs when







possible. It is recommended that conversion of programs from non -UNIX to UNIX systems use this option.

- +mm** causes each line to be moved left *n* characters, with a default value of 6. This can be useful for crunching COBOL programs.
- +pn** causes *n* blanks to be prepended (default of 6 if *n* is omitted). This option is effectively the inverse of **+mm**, and is often useful for adjusting the position of *nroff*(1) output for terminals lacking both forms tractor positioning and a settable left margin.
- +s** is used with the **-m** option of *get*(1). The **-m** option causes *get* to prepend to each generated line the appropriate SCCS SID, followed by a tab. The **+s** option causes *reform* to remove the SID from the front of the line, save it, then add it later to the end of the line. Because **+e72** is implied by this option, the effect is to produce 80-character card images with SCCS SID in columns 73-80. Up to 8 characters of the SID are shown; if it is longer, the eighth character is replaced by \* and any characters to the right of it are discarded.
- +tn** causes any line longer than *n* characters to be truncated to that length. If *n* is omitted, the length defaults to 72. Sequence numbers can thus be removed and any blanks immediately preceding them deleted.

The following illustrate typical uses of *reform*. The terms PWB and OBJECT below refer to UNIX and non-UNIX computer systems, respectively. Each arrow indicates the direction of conversion. The character ? indicates an arbitrary tab specification; see *tabs*(1) for descriptions of legal specifications.

OBJECT —> PWB (i.e., manipulation of RJE output):

Note that files transferred by RJE from OBJECT to PWB materialize with format -8.

*reform* -8 -c +t +b +i <oldfile >newfile (into COBOL)

*reform* -8 -c3 +t +m +i <oldfile >newfile (into COBOL, crunched)

NOTE: -c3 is the preferred format COBOL; it uses the least disk space of the COBOL formats.

PWB —> OBJECT (i.e., preparation of files for RJE submission):

*reform* ? -8 <oldfile >newfile (from arbitrary format into -8)

*get* -p -m sccsfile | *reform* +s | send ...

PWB ONLY (i.e., no involvement with other systems):

*pr* file | *reform* ? -0 <oldfile (print on terminal without hardware tabs)

*reform* ? -0 <oldfile >newfile (convert file to tabless format)

## DIAGNOSTICS

All diagnostics are fatal, and the offending line is displayed following the message.

"line too long" a line exceeds 512 characters (in tabless form).

"not SCCS -m" a line does not have at least one tab when **+s** flag is used.

Any of the diagnostics of *tabs*(1) can also appear.





## EXIT CODES

- 0 - normal
- 1 - any error

## SEE ALSO

get(1), nroff(1), send(1C), tabs(1), fspec(5).

## BUGS

*Reform* is aware of the meanings of backspaces and escape sequences, so that it can be used as a postprocessor for *nroff*. However, be warned that the *+e*, *+m*, and *+t* options only count characters, not positions. Anyone using these options on output containing backspaces or halfline motions will probably obtain unexpected results.

1. The first part of the document is a list of the names of the members of the committee.

2. The second part of the document is a list of the names of the members of the committee.

3. The third part of the document is a list of the names of the members of the committee.

4. The fourth part of the document is a list of the names of the members of the committee.

5. The fifth part of the document is a list of the names of the members of the committee.

6. The sixth part of the document is a list of the names of the members of the committee.

7. The seventh part of the document is a list of the names of the members of the committee.

8. The eighth part of the document is a list of the names of the members of the committee.

9. The ninth part of the document is a list of the names of the members of the committee.

10. The tenth part of the document is a list of the names of the members of the committee.

11. The eleventh part of the document is a list of the names of the members of the committee.

12. The twelfth part of the document is a list of the names of the members of the committee.

13. The thirteenth part of the document is a list of the names of the members of the committee.

14. The fourteenth part of the document is a list of the names of the members of the committee.

15. The fifteenth part of the document is a list of the names of the members of the committee.

16. The sixteenth part of the document is a list of the names of the members of the committee.

17. The seventeenth part of the document is a list of the names of the members of the committee.



## NAME

`rsh` - restricted shell (command interpreter)

## SYNOPSIS

`rsh [ flags ] [ name [ arg1 ... ] ]`

## DESCRIPTION

`Rsh` is a restricted version of the standard command interpreter `sh`(1). It is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of `rsh` are identical to those of `sh`, except that the following are disallowed:

- `cd`  
setting the value of `$PATH`  
command names containing `/`  
`>` and `>>`

When invoked with the name `-rsh`, `rsh` reads the user's `.profile` (from `$HOME/.profile`). It acts as the standard `sh` while doing this, except that an interrupt causes an immediate exit, instead of causing a return to command level. The restrictions above are enforced after `.profile` is interpreted.

When a command to be executed is found to be a shell procedure, `rsh` invokes `sh` to execute it. Thus, it is possible to provide to the end user shell procedures that have access to the full power of the standard shell, while restricting him to a limited menu of commands; this scheme assumes that the end user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the `.profile` has complete control over user actions, by performing guaranteed setup actions, then leaving the user in an appropriate directory (probably *not* the login directory).

`Rsh` is actually just a link to `sh` and any *flags* arguments are the same as for `sh`(1).

The system administrator often sets up a directory of commands that can be safely invoked by `rsh`. Some systems also provide a restricted editor `red`.

## SEE ALSO

`sh`(1), `profile`(5).





**NAME**

rxctrl - floppy disk manipulating program

**SYNOPSIS**

rxctrl *[-[fsdiz] /dev/rrx?*

**DESCRIPTION**

*Rxctrl* is used to give commands to the floppy drive:

-f Initialize data fields according to specified floppy density. See RX(4).

-s Byte swapping

-d Default: no swap of bytes

Unusual commands:

-i Interleave of sectors off.

-z Zigzag on: sequential blocks on different floppy sides.

**FILES**

/dev/rrx\*

**SEE ALSO**

rx(4), format(8), tmcctrl(1), dd(1)

11270

11270

11270

11270

11270

11270

11270

11270

11270

11270

11270

11270



## NAME

**sdiff** - side-by-side difference program

## SYNOPSIS

**sdiff** [ options ... ] file1 file2

## DESCRIPTION

*Sdiff* uses the output of *diff*(1) to produce a side-by-side listing of two files indicating those lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical, a < in the gutter if the line only exists in *file1*, a > in the gutter if the line only exists in *file2*, and a | for lines that are different.

For example:

x		y
a		a
b	<	
c	<	
d		d
	>	c

The following options exist:

**-w n** Use the next argument, *n*, as the width of the output line. The default line length is 130 characters.

**-l** Only print the left side of any lines that are identical.

**-s** Do not print identical lines.

**-o output**

Use the next argument, *output*, as the name of a third file that is created as a user controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by *diff*(1), are printed; where a set of differences share a common gutter character. After printing each set of differences, *sdiff* prompts the user with a % and waits for one of the following user-typed commands:

<b>l</b>	append the left column to the output file
<b>r</b>	append the right column to the output file
<b>s</b>	turn on silent mode; do not print identical lines
<b>v</b>	turn off silent mode
<b>e l</b>	call the editor with the left column
<b>e r</b>	call the editor with the right column
<b>e b</b>	call the editor with the concatenation of left and right
<b>e</b>	call the editor with a zero length file
<b>q</b>	exit from the program

On exit from the editor, the resulting file is concatenated on the end of the *output* file.

The first part of the document is a letter from the President of the United States to the Congress, dated January 1, 1861. It is a very important document, as it contains the President's message to the Congress at the beginning of his first term.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.

The letter is written in a very formal and dignified style, and it is one of the most important documents in the history of the United States. It contains the President's views on the state of the Union, and it is a very important document for the study of American history.



SDIFF(1)

MUNIX

SDIFF(1)

SEE ALSO  
diff(1), ed(1).

117-118-119





## NAME

sh - shell, the standard command programming language

## SYNOPSIS

sh [ -ceiknrstuvx ] [ args ]

## DESCRIPTION

*Sh* is a command programming language that executes commands read from a terminal or a file. See *Invocation* below for the meaning of arguments to the shell.

## Commands.

A *simple-command* is a sequence of non-blank words separated by blanks (a blank is a tab or a space). The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see *exec(2)*). The *value* of a simple-command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see *signal(2)* for a list of status values).

A *pipeline* is a sequence of one or more *commands* separated by |. The standard output of each command but the last is connected by a *pipe(2)* to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate.

A *list* is a sequence of one or more pipelines separated by ;, &, &&, or ||, and optionally terminated by ; or &. Of these four symbols, ; and & have equal precedence, which is lower than that of && and ||. The symbols && and || also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline; an ampersand (&) causes asynchronous execution of the preceding pipeline (i.e., the shell does *not* wait for that pipeline to finish). The symbol && (||) causes the *list* following it to be executed only if the preceding pipeline returns a zero (non-zero) exit status. An arbitrary number of new-lines may appear in a *list*, instead of semicolons, to delimit commands.

A *command* is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

**for** *name* [ *in word ...* ] **do** *list* **done**

Each time a **for** command is executed, *name* is set to the next *word* taken from the *in word* list. If *in word ...* is omitted, then the **for** command executes the *do list* once for each positional parameter that is set (see *Parameter Substitution* below). Execution ends when there are no more words in the list.

**case** *word* **in** [ *pattern* [ | *pattern* ] ... ) *list* ;; ] ... **esac**

A **case** command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for file-name generation (see *File Name Generation* below).

**if** *list* **then** *list* [ **elif** *list* **then** *list* ] ... [ **else** *list* ] **fi**

The *list* following **if** is executed and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else list** is executed. If no **else list** or **then list** is executed, then the **if** command



THE HISTORY OF THE UNITED STATES OF AMERICA

CHAPTER I

The first part of the history of the United States of America is the history of the discovery of the continent by Christopher Columbus in 1492.

The second part of the history of the United States of America is the history of the settlement of the continent by the English, French, and Spanish.

The third part of the history of the United States of America is the history of the struggle for independence from Great Britain.

The fourth part of the history of the United States of America is the history of the formation of the federal government and the establishment of the Constitution.

The fifth part of the history of the United States of America is the history of the expansion of the territory of the United States.

The sixth part of the history of the United States of America is the history of the civil war and the Reconstruction period.

The seventh part of the history of the United States of America is the history of the industrial revolution and the rise of the United States as a world power.

The eighth part of the history of the United States of America is the history of the modern period, from the end of the civil war to the present.



returns a zero exit status.

#### **while *list* do *list* done**

A **while** command repeatedly executes the **while *list*** and, if the exit status of the last command in the list is zero, executes the **do *list***; otherwise the loop terminates. If no commands in the **do *list*** are executed, then the **while** command returns a zero exit status; **until** may be used in place of **while** to negate the loop termination test.

#### **(*list*)**

Execute *list* in a sub-shell.

#### **{*list*;}**

*list* is simply executed.

The following words are only recognized as the first word of a command and when not quoted:

**if then else elif fi case esac for while until do done { }**

#### **Comments.**

A word beginning with **#** causes that word and all the following characters up to a new-line to be ignored.

#### **Command Substitution.**

The standard output from a command enclosed in a pair of grave accents (**`**) may be used as part or all of a word; trailing new-lines are removed.

#### **Parameter Substitution.**

The character **\$** is used to introduce substitutable *parameters*. Positional parameters may be assigned values by **set**. Variables may be set by writing:

**name=value [ name=value ] ...**

Pattern-matching is not performed on *value*.

#### **\${parameter}**

A *parameter* is a sequence of letters, digits, or underscores (a *name*), a digit, or any of the characters **\***, **@**, **#**, **?**, **-**, **\$**, and **!**. The value, if any, of the parameter is substituted. The braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. A *name* must begin with a letter or underscore. If *parameter* is a digit then it is a positional parameter. If *parameter* is **\*** or **@**, then all the positional parameters, starting with **\$1**, are substituted (separated by spaces). Parameter **\$0** is set from argument zero when the shell is invoked.

#### **\${parameter:-word}**

If *parameter* is set and is non-null then substitute its value; otherwise substitute *word*.

#### **\${parameter:=word}**

If *parameter* is not set or is null then set it to *word*; the value of the parameter is then substituted. Positional parameters may not be assigned to in this way.

#### **\${parameter:?word}**

If *parameter* is set and is non-null then substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, then the message "parameter null or not set" is printed.

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...



**`${parameter:+word}`**

If *parameter* is set and is non-null then substitute *word*; otherwise substitute nothing.

In the above, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, `pwd` is executed only if `d` is not set or is null:

```
echo ${d:-`pwd`}
```

If the colon (:) is omitted from the above expressions, then the shell only checks whether *parameter* is set or not.

The following parameters are automatically set by the shell:

- #** The number of positional parameters in decimal.
- Flags supplied to the shell on invocation or by the `set` command.
- ?** The decimal value returned by the last synchronously executed command.
- \$** The process number of this shell.
- !** The process number of the last background command invoked.

The following parameters are used by the shell:

- HOME** The default argument (home directory) for the `cd` command.
- PATH** The search path for commands (see *Execution* below).
- MAIL** If this variable is set to the name of a mail file, then the shell informs the user of the arrival of mail in the specified file.
- PS1** Primary prompt string, by default "**\$**".
- PS2** Secondary prompt string, by default ">".
- IFS** Internal field separators, normally space, tab, and new-line.

The shell gives default values to **PATH**, **PS1**, **PS2**, and **IFS**, while **HOME** and **MAIL** are not set at all by the shell (although **HOME** is set by `login(1)`).

**Blank Interpretation.**

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in **IFS**) and split into distinct arguments where such characters are found. Explicit null arguments (" or '') are retained. Implicit null arguments (those resulting from *parameters* that have no values) are removed.

**File Name Generation.**

Following substitution, each command *word* is scanned for the characters `*`, `?`, and `[`. If one of these characters appears then the word is regarded as a *pattern*. The word is replaced with alphabetically sorted file names that match the pattern. If no file name is found that matches the pattern, then the word is left unchanged. The character `.` at the start of a file name or immediately following a `/`, as well as the character `/` itself, must be matched explicitly.

- \*** Matches any string, including the null string.
- ?** Matches any single character.
- [...]** Matches any one of the enclosed characters. A pair of characters separated by `-` matches any character lexically

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...



execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

#### Environment.

The *environment* (see *environ(7)*) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. Executed commands inherit the same environment. If the user modifies the values of these parameters or creates new ones, none of these affects the environment unless the *export* command is used to bind the shell's parameter to the environment. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, plus any modifications or additions, all of which must be noted in *export* commands.

The environment for any *simple-command* may be augmented by prefixing it with one or more assignments to parameters. Thus:

```
TERM=450 cmd args          and
(export TERM; TERM=450; cmd args)
```

are equivalent (as far as the above execution of *cmd* is concerned).

If the *-k* flag is set, *all* keyword arguments are placed in the environment, even if they occur after the command name. The following first prints *a=b c* and then *c*:

```
echo a=b c
set -k
echo a=b c
```

#### Signals.

The *INTERRUPT* and *QUIT* signals for an invoked command are ignored if the command is followed by *&*; otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the *trap* command below).

#### Execution.

Each time a command is executed, the above substitutions are carried out. Except for the *Special Commands* listed below, a new process is created and an attempt is made to execute the command via *exec(2)*.

The shell parameter *PATH* defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is *:/bin:/usr/bin* (specifying the current directory, */bin*, and */usr/bin*, in that order). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If the command name contains a */* then the search path is not used. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not an *a.out* file, it is assumed to be a file containing shell commands. A sub-shell (i.e., a separate process) is spawned to read it. A parenthesized command is also executed in a sub-shell.

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...



## Special Commands.

The following commands are executed in the shell process and, except as specified, no input/output redirection is permitted for such commands:

- :** No effect; the command does nothing. A zero exit code is returned.
- . *file*** Read and execute commands from *file* and return. The search path specified by PATH is used to find the directory containing *file*.
- break [ *n* ]**  
Exit from the enclosing **for** or **while** loop, if any. If *n* is specified then break *n* levels.
- continue [ *n* ]**  
Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified then resume at the *n*-th enclosing loop.
- cd [ *arg* ]**  
Change the current directory to *arg*. The shell parameter HOME is the default *arg*.
- eval [ *arg* ... ]**  
The arguments are read as input to the shell and the resulting command(s) executed.
- exec [ *arg* ... ]**  
The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.
- exit [ *n* ]**  
Causes a shell to exit with the exit status specified by *n*. If *n* is omitted then the exit status is that of the last command executed (an end-of-file will also cause the shell to exit.)
- export [ *name* ... ]**  
The given *names* are marked for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, then a list of all names that are exported in this shell is printed.
- newgrp [ *arg* ... ]**  
Equivalent to **exec newgrp *arg* ....**
- read [ *name* ... ]**  
One line is read from the standard input and the first word is assigned to the first *name*, the second word to the second *name*, etc., with leftover words assigned to the last *name*. The return code is 0 unless an end-of-file is encountered.
- readonly [ *name* ... ]**  
The given *names* are marked *readonly* and the values of the these *names* may not be changed by subsequent assignment. If no arguments are given, then a list of all *readonly* names is printed.
- set [ -ekntuvx [ *arg* ... ] ]**
  - e** If the shell is non-interactive then exit immediately if a command exits with a non-zero exit status.
  - k** All keyword arguments are placed in the environment for a command, not just those that precede the command name.
  - n** Read commands but do not execute them.





- t Exit after reading and executing one command.
- u Treat unset variables as an error when substituting.
- v Print shell input lines as they are read.
- x Print commands and their arguments as they are executed.
- Do not change any of the flags; useful in setting \$1 to -.

Using + rather than - causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in \$-. The remaining arguments are positional parameters and are assigned, in order, to \$1, \$2, .... If no arguments are given then the values of all names are printed.

**shift**

The positional parameters from \$2 ... are renamed \$1 ....

**test**

Evaluate conditional expressions. See *test(1)* for usage and description.

**times**

Print the accumulated user and system times for processes run from the shell.

**trap** [ *arg* ] [ *n* ] ...

*arg* is a command to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If *arg* is absent then all trap(s) *n* are reset to their original values. If *arg* is the null string then this signal is ignored by the shell and by the commands it invokes. If *n* is 0 then the command *arg* is executed on exit from the shell. The trap command with no arguments prints a list of commands associated with each signal number.

**umask** [ *nnn* ]

The user file-creation mask is set to *nnn* (see *umask(2)*). If *nnn* is omitted, the current value of the mask is printed.

**wait**

Wait for all child processes to terminate report the termination status. If *n* is not given then all currently active child processes are waited for. The return code from this command is always zero.

**Invocation.**

If the shell is invoked through *exec(2)* and the first character of argument zero is -, commands are initially read from */etc/profile* and then from *\$HOME/.profile*, if such files exist. Thereafter, commands are read as described below, which is also the case when the shell is invoked as */bin/sh*. The flags below are interpreted by the shell on invocation only; Note that unless the -c or -s flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file:

**-c** *string*

If the -c flag is present then commands are read from *string*.

**-s**

If the -s flag is present or if no arguments remain then commands are read from the standard input. Any remaining





arguments specify the positional parameters. Shell output is written to file descriptor 2.

- i If the -i flag is present or if the shell input and output are attached to a terminal, then this shell is *interactive*. In this case TERMINATE is ignored (so that kill 0 does not kill an interactive shell) and INTERRUPT is caught and ignored (so that wait is interruptible). In all cases, QUIT is ignored by the shell.
- r If the -r flag is present the shell is a restricted shell (see *rsh(1)*).

The remaining flags and arguments are described under the *set* command above.

#### EXIT STATUS

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively then execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the *exit* command above).

#### FILES

/etc/profile  
\$HOME/.profile  
/tmp/sh\*  
/dev/null

#### SEE ALSO

cd(1), env(1), login(1), newgrp(1), rsh(1), test(1), umask(1), dup(2), exec(2), fork(2), pipe(2), signal(2), umask(2), wait(2), a.out(5), profile(5), environ(7).

#### BUGS

The command *readonly* (without arguments) produces the same output as the command *export*.

If << is used to provide standard input to an asynchronous process invoked by &, the shell gets mixed up about naming the input document; a garbage file /tmp/sh\* is created and the shell complains about not being able to find that file by another name.

...the ...  
...the ...  
...the ...  
...the ...  
...the ...

...the ...

...the ...  
...the ...  
...the ...  
...the ...

...  
...  
...  
...

...the ...  
...the ...

...the ...  
...the ...  
...the ...  
...the ...  
...the ...



**NAME**

size — size of an object file

**SYNOPSIS**

size [ object ... ]

**DESCRIPTION**

*Size* prints the (decimal) number of bytes required by the text, data, bss and stack portions, and their sum in octal and decimal, of each object-file argument. If no file is specified, **a.out** is used.

**SEE ALSO**

**a.out(5)**





## NAME

stctrl - special streamer features  
stskip - skip files

## SYNOPSIS

stctrl [-e] [-r]  
stskip count

## DESCRIPTION

*Stctrl* is used to execute special streamer commands. The possible options are:

-e to erase the whole tape.

-r to make a retension of the tape.

*Stskip* skips the next files on the tape. Count holds the number of files to be skipped.

These two commands always refer to streamer drive 0.

## FILES

/dev/rst0  
/dev/nrst0

## SEE ALSO

st(4)





## NAME

*strip* - remove symbols and relocation bits

## SYNOPSIS

*strip* name ...

## DESCRIPTION

*Strip* removes the symbol table and relocation bits ordinarily attached to the output of the assembler and link editor. This is useful to save space after a program has been debugged.

The effect of *strip* is the same as use of the *-s* option of *ld*.

If *name* is an archive file, *strip* will remove the local symbols from any *a.out* format files it finds in the archive. Certain libraries, such as those residing in */lib*, have no need for local symbols. By deleting them, the size of the archive is decreased and link editing performance is increased.

## FILES

*/tmp/stm\** temporary file

## SEE ALSO

*ld*(1).

MEMORANDUM FOR THE RECORD

10/1/77

10/1/77

10/1/77

On 10/1/77, the following information was received from the [redacted] regarding the [redacted] of the [redacted] in the [redacted] area.

The [redacted] of the [redacted] is [redacted] and is [redacted] in the [redacted] area.

The [redacted] of the [redacted] is [redacted] and is [redacted] in the [redacted] area. The [redacted] of the [redacted] is [redacted] and is [redacted] in the [redacted] area. The [redacted] of the [redacted] is [redacted] and is [redacted] in the [redacted] area.

Very truly yours,

10/1/77

10/1/77



## NAME

**stty** – set terminal options of the current output terminal  
**setty** – set terminal options of an other output terminal

## SYNOPSIS

**stty** [ option ... ]  
**setty** special [ option ... ]

## DESCRIPTION

**Stty** sets certain I/O options on the current output terminal.

**Setty** does the same for the terminal named *special*.

With no argument, they report the current settings of the options. The option strings are selected from the following set:

**even** allow even parity  
**–even** disallow even parity  
**odd** allow odd parity  
**–odd** disallow odd parity  
**raw** raw mode input (no erase, kill, interrupt, quit, EOT; parity bit passed back)  
**–raw** negate raw mode  
**cooked**  
     same as '–raw'  
**cbreak** make each character available to *read(2)* as received; no erase and kill  
**–cbreak**  
     make characters available to *read* only when newline is received  
**–nl** allow carriage return for new-line, and output CR-LF for carriage return or new-line  
**nl** accept only new-line to end lines  
**echo** echo back every character typed  
**–echo** do not echo characters  
**lcase** map upper case to lower case  
**–lcase** do not map case  
**–tabs** replace tabs by spaces when printing  
**tabs** preserve tabs  
**ek** reset erase and kill characters back to normal # and @  
**erase c**  
     set erase character to c. C can be of the form '^X' which is interpreted as a 'control X'.  
**kill c** set kill character to c. '^X' works here also.  
**cr0 cr1 cr2 cr3**  
     select style of delay for carriage return (see *ioctl(2)*)  
**nl0 nl1 nl2 nl3**  
     select style of delay for linefeed  
**tab0 tab1 tab2 tab3**  
     select style of delay for tab  
**ff0 ff1** select style of delay for form feed  
**bs0 bs1**  
     select style of delay for backspace  
**tty33** set all modes suitable for the Teletype Corporation Model 33 terminal.  
**tty37** set all modes suitable for the Teletype Corporation Model 37 terminal.

1911-12

...

...

...

...

...

...

...

...

...



**vt05** set all modes suitable for Digital Equipment Corp. VT05 terminal  
**tn300** set all modes suitable for a General Electric TermiNet 300  
**ti700** set all modes suitable for Texas Instruments 700 series terminal  
**tek** set all modes suitable for Tektronix 4014 terminal  
**hup** hang up dataphone on last close.  
**-hup** do not hang up dataphone on last close.  
**0** hang up phone line immediately  
**50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 exta extb**  
Set terminal baud rate to the number given, if possible. (These are the speeds supported by the DH-11 interface).

## SEE ALSO

**ioctl(2), tabs(1)**





## NAME

`tbl` - format tables for `nroff` or `troff`

## SYNOPSIS

`tbl [ -TX ] [ files ]`

## DESCRIPTION

`Tbl` is a preprocessor that formats tables for `nroff(1)` or `troff(1)`. The input files are copied to the standard output, except for lines between `.TS` and `.TE` command lines, which are assumed to describe tables and are re-formatted by `tbl`. (The `.TS` and `.TE` command lines are not altered by `tbl`).

`.TS` is followed by global options. The available global options are:

- `center` center the table (default is left-adjust);
- `expand` make the table as wide as the current line length;
- `box` enclose the table in a box;
- `doublebox` enclose the table in a double box;
- `allbox` enclose each item of the table in a box;
- `tab (x)` use the character `x` instead of a tab to separate items in a line of input data.

The global options, if any, are terminated with a semi-colon (;).

Next come lines describing the format of each line of the table. Each such format line describes one line of the actual table, except that the last format line (which must end with a period) describes *all* remaining lines of the table. Each column of each line of the table is described by a single key-letter, optionally followed by specifiers that determine the font and point size of the corresponding item, that indicate where vertical bars are to appear between columns, that determine column width, inter-column spacing, etc. The available key-letters are:

- `c` center item within the column;
- `r` right-adjust item within the column;
- `l` left-adjust item within the column;
- `n` numerically adjust item in the column: units positions of numbers are aligned vertically;
- `s` span previous item on the left into this column;
- `a` center longest line in this column and then left-adjust all other lines in this column with respect to that centered line;
- `^` span down previous entry in this column;
- `_` replace this entry with a horizontal line;
- `=` replace this entry with a double horizontal line.

The characters `B` and `I` stand for the bold and italic fonts, respectively; the character `|` indicates a vertical line between columns.

The format lines are followed by lines containing the actual data for the table, followed finally by `.TE`. Within such data lines, data items are normally separated by tab characters.

If a data line consists of only `_` or `=`, a single or double line, respectively, is drawn across the table at that point; if a *single item* in a data line consists of only `_` or `=`, then that item is replaced by a single or double line.

100

100

100

100

100

100

100

100

100

100

100

100

100

100



Full details of all these and other features of *tbl* are given in the reference manual cited below.

The *-TX* option forces *tbl* to use only full vertical line motions, making the output more suitable for devices that cannot generate partial vertical line motions (e.g., line printers).

If no file names are given as arguments, *tbl* reads the standard input, so it may be used as a filter. When it is used with *eqn(1)* or *neqn(1)*, *tbl* should come first to minimize the volume of data passed through pipes.

#### EXAMPLE

If we let  $\rightarrow$  represent a tab (which should be typed as a genuine tab), then the input:

```
.TS
center box ;
cB s s
cI | cI s
^ | c c
l | n n .
Household Population
=
Town→Households
→Number→Size
=
Bedminster→789→3.26
Bernards Twp.→3087→3.74
Bernardsville→2018→3.30
Bound Brook→3425→3.04
Bridgewater→7897→3.81
Far Hills→240→3.19
.TE
```

yields:

Household Population		
Town	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Bridgewater	7897	3.81
Far Hills	240	3.19

#### SEE ALSO

*TBL-A Program to Format Tables* by M. E. Lesk  
*eqn(1)*, *mm(1)*, *mmnt(1)*, *troff(1)*, *mm(7)*, *mv(7)*.

#### BUGS

See *BUGS* under *troff(1)*.

The first part of the report is a general description of the project. It includes a brief history of the project, a statement of the problem, and a description of the objectives of the project. The second part of the report is a detailed description of the methodology used in the project. It includes a description of the data collection methods, a description of the data analysis methods, and a description of the results of the project.

The third part of the report is a discussion of the results of the project. It includes a discussion of the findings of the project, a discussion of the implications of the findings, and a discussion of the limitations of the project. The fourth part of the report is a conclusion and a list of references.

The fifth part of the report is a list of references. It includes a list of books, a list of articles, and a list of other sources. The sixth part of the report is a list of appendices. It includes a list of tables, a list of figures, and a list of other appendices.

The seventh part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices. The eighth part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices.

The ninth part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices. The tenth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices.

The eleventh part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices. The twelfth part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices.

The thirteenth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices. The fourteenth part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices.

The fifteenth part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices. The sixteenth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices.

The seventeenth part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices. The eighteenth part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices.

The nineteenth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices. The twentieth part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices.

The twenty-first part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices. The twenty-second part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices.

The twenty-third part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices. The twenty-fourth part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices.

The twenty-fifth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices. The twenty-sixth part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices.

The twenty-seventh part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices. The twenty-eighth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices.

The twenty-ninth part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices. The thirtieth part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices.

The thirty-first part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices. The thirty-second part of the report is a list of figures. It includes a list of tables, a list of figures, and a list of other appendices.

The thirty-third part of the report is a list of other appendices. It includes a list of tables, a list of figures, and a list of other appendices. The thirty-fourth part of the report is a list of tables. It includes a list of tables, a list of figures, and a list of other appendices.



## NAME

*tmctrl* - magnetic tape manipulating program  
*tmskip* - skip files

## SYNOPSIS

*tmctrl* [-sd] [tapename]  
*tmskip* count

## DESCRIPTION

*Tmctrl* is used to give commands to the tape drive. If no tapename is specified, /dev/rmt0 is used.

Here are the commands:

- s    byte swapping
- d    default: no swap of bytes

*Tmskip* skips the next *count* files of the tape.

## FILES

/dev/rmt\* /dev/nrmt\*    Raw magnetic tape interface

## SEE ALSO

tm(4)

1900

1900

1900

1900

1900

1900

1900

1900

1900

1900

1900

1900



## NAME

tplot - graphics filters

## SYNOPSIS

tplot [ -Tterminal [ -e raster ] ]

## DESCRIPTION

These commands read plotting instructions (see *plot(5)*) from the standard input and in general produce, on the standard output, plotting instructions suitable for a particular *terminal*. If no *terminal* is specified, the environment parameter **\$TERM** (see *environ(7)*) is used. Known *terminals* are:

300 DASI 300.

300S DASI 300s.

450 DASI 450.

4014 Tektronix 4014.

lbp Canon lbp10 laserbeam printer

ver Versatec D1200A. This version of *plot* places a scan-converted image in **/usr/tmp/raster\$\$** and sends the result directly to the plotter device, rather than to the standard output. The **-e** option causes a previously scan-converted file *raster* to be sent to the plotter.

## FILES

**/usr/lib/t300**

**/usr/lib/t300s**

**/usr/lib/t450**

**/usr/lib/t4014**

**/usr/lib/lplot**

**/usr/lib/vplot**

**/usr/tmp/raster\$\$**

## SEE ALSO

*plot(3X)*, *plot(5)*, *term(7)*.





## NAME

`troff`, `nroff` - typeset or format text

## SYNOPSIS

`nroff` [ options ] [ files ]

`troff` [ options ] [ files ]

## DESCRIPTION

*Nroff* formats text contained in *files* (standard input by default) for printing on typewriter-like devices and line printers; similarly, *troff* formats text for a Wang Laboratories, Inc., C/A/T phototypesetter. Their capabilities are described in the *NROFF/TROFF User's Manual* cited below.

An argument consisting of a minus (-) is taken to be a file name corresponding to the standard input. The *options*, which may appear in any order, but must appear before the *files*, are:

- olist Print only pages whose page numbers appear in the *list* of numbers and ranges, separated by commas. A range *N-M* means pages *N* through *M*; an initial -*N* means from the beginning to page *N*; and a final *N-* means from *N* to the end. (See *BUGS* below.)
- n*N* Number first generated page *N*.
- s*N* Stop every *N* pages. *Nroff* will halt *after* every *N* pages (default *N*=1) to allow paper loading or changing, and will resume upon receipt of a line-feed or new-line (new-lines do not work in pipelines, e.g., with *mm*(1)). This option does not work if the output of *nroff* is piped through *col*(1). *Troff* will stop the phototypesetter every *N* pages, produce a trailer to allow changing cassettes, and resume when the typesetter's start button is pressed. When *nroff* (*troff*) halts between pages, an ASCII BEL (in *troff*, the message *page stop*) is sent to the terminal.
- ra*N* Set register *a* (which must have a one-character name) to *N*.
- i Read standard input after *files* are exhausted.
- q Invoke the simultaneous input-output mode of the *.rd* request.
- z Print only messages generated by *.tm* (terminal message) requests.
- m*name* Prepend to the input *files* the non-compacted (ASCII text) macro file */usr/lib/tmac/tmac.name*.
- c*name* Prepend to the input *files* the compacted macro files */usr/lib/macros/cmp.[nt].[dt].name* and */usr/lib/macros/ucmp.[nt].name*.
- k*name* Compact the macros used in this invocation of *nroff* / *troff*, placing the output in files *[dt].name* in the current directory (see the May 1979 Addendum to the *NROFF/TROFF User's Manual* for details of compacting macro files).

## Nroff only:

- T*name* Prepare output for specified terminal. Known *names* are 37 for the (default) TELETYPE® Model 37 terminal, tn300 for the GE TermiNet 300 (or any terminal without half-line capability), 300s for the DASI 300s, 300 for the DASI 300, 450 for the DASI 450, lp for a (generic) ASCII line printer, 382 for the DTC-382, 4000A for the Trendata 4000A, 832 for the Anderson Jacobson

1914

1915

1916

1917

1918

1919

1920

1921

1922

1923

1924

1925

1926

1927

1928

1929

1930

1931

1932

1933

1934

1935

1936

1937



- 832, X for a (generic) EBCDIC printer, and 2631 for the Hewlett Packard 2631 line printer.
- e Produce equally-spaced words in adjusted lines, using the full resolution of the particular terminal.
  - h Use output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every 8 nominal character widths.
  - un Set the emboldening factor (number of character overstrikes) for the third font position (bold) to *n*, or to zero if *n* is missing.

## Troff only:

- t Direct output to the standard output instead of the phototypesetter.
- f Refrain from feeding out paper and stopping phototypesetter at the end of the run.
- w Wait until phototypesetter is available, if it is currently busy.
- b Report whether the phototypesetter is busy or available. No text processing is done.
- a Send a printable ASCII approximation of the results to the standard output.
- p*N* Print all characters in point size *N* while retaining all prescribed spacings and motions, to reduce phototypesetter elapsed time.
- g Prepare output for the Murray Hill Computation Center phototypesetter and direct it to the standard output (see *gcat(1C)*). This option is not compatible with the *-s* option; furthermore, when this option is invoked, all *.fp* (font position) requests (if any) in the *troff* input must come before the first break, and no *.tl* requests may come before the first break.
- T*name* Use font-width tables for device *name* (the font tables are found in */usr/lib/font/name/\**). Currently, no *names* are supported.

## FILES

<i>/usr/lib/suftab</i>	suffix hyphenation tables
<i>/tmp/ta\$#</i>	temporary file
<i>/usr/lib/tmac/tmac.*</i>	standard macro files and pointers
<i>/usr/lib/macros/*</i>	standard macro files
<i>/usr/lib/term/*</i>	terminal driving tables for <i>nroff</i>
<i>/usr/lib/font/*</i>	font width tables for <i>troff</i>

## SEE ALSO

*NROFF/TROFF User's Manual* by J. F. Ossanna.  
*A TROFF Tutorial* by B. W. Kernighan.  
*eqn(1)*, *tbl(1)*, *mm(7)*.  
*col(1)*, *greek(1)*, *mm(1)* (*nroff* only).  
*gcat(1C)*, *mmt(1)*, *tc(1)*, *mv(7)* (*troff* only).

## BUGS

*Nroff/troff* believes in Eastern Standard Time; as a result, depending on the time of the year and on your local time zone, the date that *nroff/troff* generates may be off by one day from your idea of what the date is.

When *nroff/troff* is used with the *-olist* option inside a pipeline (e.g.,





with one or more of *cw*(1), *eqn*(1), and *tbl*(1)), it may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

THE S. S. HOLLANDER, DEPT. OF COMMERCE, BUREAU OF MARITIME SERVICE, WASHINGTON, D. C.  
May 11, 1944. Mr. J. H. [illegible] [illegible] [illegible] [illegible]



## NAME

uname - print name of current UNIX

## SYNOPSIS

uname [ -snrva ]

## DESCRIPTION

*Uname* prints the current system name of UNIX on the standard output file. It is mainly useful to determine what system one is using. The options cause selected information returned by *uname(2)* to be printed:

- s    print the system name (default).
- n    print the nodename (the nodename may be a name that the system is known by to a communications network).
- r    print the operating system release.
- v    print the operating system version.
- a    print all the above information.

## SEE ALSO

uname(2).

1. The first part of the report is a general statement of the purpose and scope of the study. It is followed by a description of the methods used in the investigation. The results of the study are then presented in a series of tables and figures. The final part of the report is a discussion of the results and a conclusion.

2. The second part of the report is a detailed description of the methods used in the investigation. It includes a description of the experimental apparatus, the procedures used in the experiments, and the methods used in the analysis of the data.

3. The third part of the report is a presentation of the results of the study. It includes a series of tables and figures that show the data obtained from the experiments. The results are then discussed in a series of paragraphs that explain the significance of the findings.

4. The fourth part of the report is a discussion of the results and a conclusion. It includes a summary of the findings and a discussion of their implications. The report concludes with a statement of the author's conclusions and a list of references.



**NAME**

**uuclean** - uucp spool directory clean-up

**SYNOPSIS**

**uuclean** [ options ] ...

**DESCRIPTION**

*Uuclean* will scan the spool directory for files with the specified prefix and delete all those which are older than the specified number of hours.

The following options are available.

**-ddirectory**

Clean *directory* instead of the spool directory.

**-ppre** Scan for files with *pre* as the file prefix. Up to 10 **-p** arguments may be specified. A **-p** without any *pre* following will cause all files older than the specified time to be deleted.

**-ntime**

Files whose age is more than *time* hours will be deleted if the prefix test is satisfied. (default time is 72 hours)

**-m** Send mail to the owner of the file when it is deleted.

This program will typically be started by *cron*(1M).

**FILES**

/usr/lib/uucp

directory with commands used by *uuclean* internally

/usr/spool/uucp

spool directory

**SEE ALSO**

*uucp*(1C), *uux*(1C).

The first of the three main parts of the book is devoted to a general survey of the history of the world from the beginning of time to the present day. The second part is devoted to a detailed study of the history of the United States from the time of the first settlement to the present day. The third part is devoted to a detailed study of the history of the United States from the time of the first settlement to the present day.

The first of the three main parts of the book is devoted to a general survey of the history of the world from the beginning of time to the present day. The second part is devoted to a detailed study of the history of the United States from the time of the first settlement to the present day. The third part is devoted to a detailed study of the history of the United States from the time of the first settlement to the present day.

The first of the three main parts of the book is devoted to a general survey of the history of the world from the beginning of time to the present day. The second part is devoted to a detailed study of the history of the United States from the time of the first settlement to the present day. The third part is devoted to a detailed study of the history of the United States from the time of the first settlement to the present day.

Continued on next page



## NAME

**uucp, uulog, uuname** — unix to unix copy

## SYNOPSIS

**uucp** [ option ] ... *source-file* ... *destination-file*

**uulog** [ option ] ...

**uuname**

## DESCRIPTION

*Uucp* copies files named by the *source-file* arguments to the *destination-file* argument. A file name may be a path name on your machine, or may have the form:

*system-name!path-name*

where *system-name* is taken from a list of system names which *uucp* knows about. Shell metacharacters *?\*[]* appearing in *path-name* will be expanded on the appropriate system.

Path names may be one of:

- (1) a full path name;
- (2) a path name preceded by *~user* where *user* is a login name on the specified system and is replaced by that user's login directory;
- (3) a path name preceded by *~/user* where *user* is a login name on the specified system and is replaced by that user's directory under PUBDIR;
- (4) anything else is prefixed by the current directory.

If the result is an erroneous path name for the remote system the copy will fail. If the *destination-file* is a directory, the last part of the *source-file* name is used.

*Uucp* preserves execute permissions across the transmission and gives 0666 read and write permissions (see *chmod(2)*).

The following options are interpreted by *uucp*:

- d** Make all necessary directories for the file copy (default).
- f** Do not make intermediate directories for the file copy.
- c** Use the source file when copying out rather than copying the file to the spool directory (default).
- C** Copy the source file to the spool directory.
- m** Send mail to the requester when the copy is complete.
- nuser**  
Notify *user* on the remote system that a file was sent.
- esys**  
Send the *uucp* command to system *sys* to be executed there. (Note — this will only be successful if the remote machine allows the *uucp* command to be executed by */usr/lib/uucp/uuxqt*.)

*Uulog* maintains a summary log of *uucp* and *uux(1C)* transactions in the file */usr/spool/uucp/LOGFILE* by gathering information from partial log files named */usr/spool/uucp/LOG.\*?*. (These files will only be created if

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100



the LOGFILE is being used by another process.) It removes the partial log files.

The options cause *uulog* to print logging information:

**-ssys**

Print information about work involving system *sys*.

**-uuser**

Print information about work done for the specified *user*.

*Uuname* lists the uucp names of known systems. The **-l** option returns the local system name.

#### FILES

*/usr/spool/uucp*

spool directory

*/usr/spool/uucppublic*

public directory for receiving and sending (PUBDIR)

*/usr/lib/uucp/\**

other data and program files

#### SEE ALSO

*mail(1)*, *uux(1C)*.

*Uucp Implementation Description* by D. A. Nowitz.

#### WARNING

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by path name; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary path names. As distributed, the remotely accessible files are those whose names begin */usr/spool/uucppublic* (equivalent to *~nuucp* or just *~*).

#### BUGS

All files received by *uucp* will be owned by *uucp*.

The **-m** option will only work sending files or receiving a single file. (Receiving multiple files specified by special shell characters *?\*[]* will not activate the **-m** option.)

The first part of the report is a summary of the work done during the year.

The second part is a detailed account of the work done during the year.

The third part is a summary of the work done during the year.

The fourth part is a summary of the work done during the year.

The fifth part is a summary of the work done during the year.

The sixth part is a summary of the work done during the year.

The seventh part is a summary of the work done during the year.

The eighth part is a summary of the work done during the year.

The ninth part is a summary of the work done during the year.

The tenth part is a summary of the work done during the year.

The eleventh part is a summary of the work done during the year.

The twelfth part is a summary of the work done during the year.

The thirteenth part is a summary of the work done during the year.

The fourteenth part is a summary of the work done during the year.

The fifteenth part is a summary of the work done during the year.

The sixteenth part is a summary of the work done during the year.



## NAME

`uustat` - `uucp` status inquiry and job control

## SYNOPSIS

`uustat` [ option ] ...

## DESCRIPTION

`Uustat` will display the status of, or cancel, previously specified `uucp` commands, or provide general status on `uucp` connections to other systems. The following options are recognized:

- `mmch` Report the status of accessibility of machine *mch*. If *mch* is specified as `all`, then the status of all machines known to the local `uucp` are provided.
- `kjobn` Kill the `uucp` request whose job number is *jobn*. The killed `uucp` request must belong to the person issuing the `uustat` command unless he is the super-user.
- `chour` Remove the status entries which are older than *hour* hours. This administrative option can only be initiated by the user `uucp` or the super-user.
- `uuser` Report the status of all `uucp` requests issued by *user*.
- `ssys` Report the status of all `uucp` requests which communicate with remote system *sys*.
- `ohour` Report the status of all `uucp` requests which are older than *hour* hours.
- `yhour` Report the status of all `uucp` requests which are younger than *hour* hours.
- `jall` Report the status of all the `uucp` requests.
- `v` Report the `uucp` status verbosely. If this option is not specified, a status code is printed with each `uucp` request.

When no options are given, `uustat` outputs the status of all `uucp` requests issued by the current user. Note that only one of the options `-j`, `-m`, `-k`, `-c`, or the rest of other options may be specified.

For example, the command

`uustat -uhdc -smhtsa -y72 -v`

will print the verbose status of all `uucp` requests that were issued by user *hdc* to communicate with system *mhtsa* within the last 72 hours. The meanings of the job request status are:

job-number user remote-system command-time status-time status  
where the *status* may be either an octal number or a verbose description. The octal code corresponds to the following description:

OCTAL	STATUS
00001	the copy failed, but the reason cannot be determined
00002	permission to access local file is denied
00004	permission to access remote file is denied
00010	bad <code>uucp</code> command is generated
00020	remote system cannot create temporary file
00040	cannot copy to remote directory
00100	cannot copy to local directory
00200	local system cannot create temporary file





00400 cannot execute *uucp*  
01000 copy succeeded  
02000 copy finished, job deleted  
04000 job is queued

The meanings of the machine accessibility status are:

system-name time status

where *time* is the latest status time and *status* is a self-explanatory description of the machine status.

#### FILES

/usr/spool/uucp  
                    pool directory  
/usr/lib/uucp/L\_stat  
                    system status file  
/usr/lib/uucp/R\_stat  
                    request status file

#### SEE ALSO

*uucp(1C)*.  
*Uustat - A UUCP Status Inquiry Program*, by H. Che.

THE STATE OF TEXAS  
COUNTY OF DALLAS  
I, the undersigned, Clerk of the County of Dallas, Texas, do hereby certify that the within and foregoing is a true and correct copy of the original as the same appears in the records of the County of Dallas, Texas.

Witness my hand and the seal of the County of Dallas, Texas, this 1st day of January, 2000.

\_\_\_\_\_  
Clerk of the County of Dallas, Texas

NOTARY PUBLIC



## NAME

uusub - monitor uucp network

## SYNOPSIS

uusub [ options ]

## DESCRIPTION

*uusub* defines a *uucp* subnetwork and monitors the connection and traffic among the members of the subnetwork. The following options are available:

- asys* Add *sys* to the subnetwork.
- dsys* Delete *sys* from the subnetwork.
- l* Report the statistics on connections.
- r* Report the statistics on traffic amount.
- f* Flush the connection statistics.
- uhr* Gather the traffic statistics over the past *hr* hours.
- csys* Exercise the connection to the system *sys*. If *sys* is specified as *all*, then exercise the connection to all the systems in the subnetwork.

The meanings of the connections report are:

*sys* #*call* #*ok* *time* #*dev* #*login* #*nack* #*other*

where *sys* is the remote system name, #*call* is the number of times the local system tries to call *sys* since the last flush was done, #*ok* is the number of successful connections, *time* is the latest successful connect time, #*dev* is the number of unsuccessful connections because of no available device (e.g. ACU), #*login* is the number of unsuccessful connections because of login failure, #*nack* is the number of unsuccessful connections because of no response (e.g. line busy, system down), and #*other* is the number of unsuccessful connections because of other reasons.

The meanings of the traffic statistics are:

*sfile* *sbyte* *rfile* *rbyte*

where *sfile* is the number of files sent and *sbyte* is the number of bytes sent over the period of time indicated in the latest *uusub* command with the *-uhr* option. Similarly, *rfile* and *rbyte* are the numbers of files and bytes received.

The command:

**uusub -c all -u 24**

is typically started by *cron*(1M) once a day.

## FILES

/usr/spool/uucp/SYSLOG  
                                  system log file  
/usr/lib/uucp/L\_sub  
                                  connection statistics  
/usr/lib/uucp/R\_sub  
                                  traffic statistics

## SEE ALSO

uucp(1C), uustat(1C).

The following information was obtained from the records of the Department of the Interior, Bureau of Land Management, for the year ending December 31, 1964.

The total number of acres of land owned by the United States is 1,000,000,000. The total number of acres of land owned by the State of California is 100,000,000. The total number of acres of land owned by the State of Texas is 100,000,000.

The following table shows the number of acres of land owned by the United States, the State of California, and the State of Texas for the years 1960, 1961, 1962, 1963, and 1964.

Year	United States	State of California	State of Texas
1960	1,000,000,000	100,000,000	100,000,000
1961	1,000,000,000	100,000,000	100,000,000
1962	1,000,000,000	100,000,000	100,000,000
1963	1,000,000,000	100,000,000	100,000,000
1964	1,000,000,000	100,000,000	100,000,000

The following table shows the number of acres of land owned by the United States, the State of California, and the State of Texas for the years 1960, 1961, 1962, 1963, and 1964.

Year	United States	State of California	State of Texas
1960	1,000,000,000	100,000,000	100,000,000
1961	1,000,000,000	100,000,000	100,000,000
1962	1,000,000,000	100,000,000	100,000,000
1963	1,000,000,000	100,000,000	100,000,000
1964	1,000,000,000	100,000,000	100,000,000

The following table shows the number of acres of land owned by the United States, the State of California, and the State of Texas for the years 1960, 1961, 1962, 1963, and 1964.

Year	United States	State of California	State of Texas
1960	1,000,000,000	100,000,000	100,000,000
1961	1,000,000,000	100,000,000	100,000,000
1962	1,000,000,000	100,000,000	100,000,000
1963	1,000,000,000	100,000,000	100,000,000
1964	1,000,000,000	100,000,000	100,000,000

The following table shows the number of acres of land owned by the United States, the State of California, and the State of Texas for the years 1960, 1961, 1962, 1963, and 1964.

Year	United States	State of California	State of Texas
1960	1,000,000,000	100,000,000	100,000,000
1961	1,000,000,000	100,000,000	100,000,000
1962	1,000,000,000	100,000,000	100,000,000
1963	1,000,000,000	100,000,000	100,000,000
1964	1,000,000,000	100,000,000	100,000,000



## NAME

uuto, uupick - public UNIX-to-UNIX file copy

## SYNOPSIS

uuto [ options ] source-files destination  
uupick [ -s system ]

## DESCRIPTION

*Uuto* sends *source-files* to *destination*. *Uuto* uses the *uucp*(1C) facility to send files, while it allows the local system to control the file access. A source-file name is a path name on your machine. Destination has the form:

system#user

where *system* is taken from a list of system names that *uucp* knows about (see *uname*(1C)). *Logname* is the login name of someone on the specified system.

Two options are available:

- p Copy the source file into the spool directory before transmission.
- m Send mail to the sender when the copy is complete.

The files (or sub-trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is a public directory defined in the *uucp* source. Specifically the files are sent to

PUBDIR/receive/user/mysystem/files.

The destined recipient is notified by *mail*(1) of the arrival of files.

*Uupick* accepts or rejects the files transmitted to the user. Specifically, *uupick* searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from system: [file file-name] [dir dirname] ?

*Uupick* then reads a line from the standard input to determine the disposition of the file:

- <new-line> Go on to next entry.
- d Delete the entry.
- m [ dir ] Move the entry to named directory *dir* (current directory is default).
- a [ dir ] Same as m except moving all the files sent from *system*.
- p Print the content of the file.
- q Stop.
- EOT (control-d) Same as q.
- !command Escape to the shell to do *command*.
- Print a command summary.

*Uupick* invoked with the *-ssystem* option will only search the PUBDIR for files sent from *system*.

1. The first part of the document discusses the importance of maintaining accurate records.

2. It then goes on to describe the various methods used to collect and analyze data.

3. The third section details the results of the experiments and the conclusions drawn from them.

4. Finally, the document discusses the implications of the findings and suggests areas for further research.

5. The last part of the document provides a summary of the key points and a list of references.

6. The document is written in a clear and concise style, making it easy to read and understand.

7. It is a valuable resource for anyone interested in the field of research and data analysis.

8. The document is well-organized and easy to navigate, with clear headings and subheadings.

9. It provides a comprehensive overview of the topic and is a great starting point for further exploration.

10. The document is a high-quality piece of work and is highly recommended for anyone in the field.

11. It is a well-written and informative document that provides a clear and concise overview of the topic.

12. The document is a valuable resource for anyone interested in the field of research and data analysis.

13. It is a well-organized and easy to navigate document, with clear headings and subheadings.

14. The document provides a comprehensive overview of the topic and is a great starting point for further exploration.

15. It is a high-quality piece of work and is highly recommended for anyone in the field.

16. The document is a well-written and informative document that provides a clear and concise overview of the topic.

17. It is a valuable resource for anyone interested in the field of research and data analysis.

18. The document is a well-organized and easy to navigate document, with clear headings and subheadings.



UUTO(1C)

MUNDX

UUTO(1C)

FILES

PUBDIR      /usr/spool/uucppublic      public directory

SEE ALSO

mail(1), uuclean(1M), uucp(1C), uulog(1C), uuname(1C), uustat(1C),  
uux(1C).

1900

1901

1902

1903

1904

1905



## NAME

`uux` - unix to unix command execution

## SYNOPSIS

`uux [ - ] command-string`

## DESCRIPTION

`Uux` will gather zero or more files from various systems, execute a command on a specified system and then send standard output to a file on a specified system. Note that, for security reasons, many installations will limit the list of commands executable on behalf of an incoming request from `uux`. Many sites will permit little more than the receipt of mail (see `mail(1)`) via `uux`.

The *command-string* is made up of one or more arguments that look like a Shell command line, except that the command and file names may be prefixed by *system-name*!. A null *system-name* is interpreted as the local system.

File names may be one of

- (1) a full path name;
- (2) a path name preceded by `~xxx` where `xxx` is a login name on the specified system and is replaced by that user's login directory;
- (3) anything else is prefixed by the current directory.

The `-` option will cause the standard input to the `uux` command to be the standard input to the *command-string*. For example, the command

```
uux "!.diff usg!/usr/dan/f1 pwba!/a4/dan/f1 > !f1.diff"
```

will get the `f1` files from the "usg" and "pwba" machines, execute a *diff* command and put the results in `f1.diff` in the local directory.

Any special shell characters such as `<>|` should be quoted either by quoting the entire *command-string*, or quoting the special characters as individual arguments.

`Uux` will attempt to get all files to the execution system. For files which are output files, the file name must be escaped using parentheses. For example, the command

```
uux a!uucp b!/usr/file \ (c!/usr/file\)
```

will send a `uucp` command to system "a" to get `/usr/file` from system "b" and send it to system "c".

`Uux` will notify you if the requested command on the remote system was disallowed. The response comes by remote mail from the remote machine.

## FILES

`/usr/lib/uucp/spool`                      spool directory  
`/usr/lib/uucp/*`                      other data and programs

## SEE ALSO

`uuclean(1M)`, `uucp(1C)`.  
*Uucp Implementation Description* by D. A. Nowitz

1. The first part of the report deals with the general situation of the country and the progress of the work during the year. It is divided into two main sections: the first section deals with the general situation and the second section deals with the progress of the work.

2. The second part of the report deals with the results of the work during the year. It is divided into two main sections: the first section deals with the results of the work in the field and the second section deals with the results of the work in the laboratory.

3. The third part of the report deals with the conclusions of the work during the year. It is divided into two main sections: the first section deals with the conclusions of the work in the field and the second section deals with the conclusions of the work in the laboratory.

4. The fourth part of the report deals with the recommendations of the work during the year. It is divided into two main sections: the first section deals with the recommendations of the work in the field and the second section deals with the recommendations of the work in the laboratory.

5. The fifth part of the report deals with the summary of the work during the year. It is divided into two main sections: the first section deals with the summary of the work in the field and the second section deals with the summary of the work in the laboratory.



## BUGS

Only the first command of a shell pipeline may have a *system-name*!. All other commands are executed on the system of the first command. The use of the shell metacharacter *\** will probably not do what you want it to do. The shell tokens *<<* and *>>* are not implemented.

1. The first part of the document is a letter from the President of the United States to the Congress, dated January 1, 1861. It is a very important document, as it contains the President's message to the Congress at the beginning of his first term. The letter is written in a formal, dignified style, and it is one of the most important documents in American history.



## NAME

**volcopy, labelit** - copy file systems with label checking

## SYNOPSIS

**volcopy** [-S][-bpibits-per-inch ] [-feetsize ] *fsname* *special1* *volname1*  
*special2* *volname2*

**labelit** *special* [ *fsname* *volume* [ -n ] ]

## DESCRIPTION

*Volcopy* makes a literal copy of the file system using a blocksize matched to the device (10 blocks for 800/1600 bpi tape; 110 blocks for cartridge tapes if -S option used; 88 blocks for everything else). Using *volcopy*, a 2400 foot/1600 bpi tape will hold a 65K file system. The optional flag arguments are used only with tapes (-bpi -- bits-per-inch; -feet -- size of reel in feet). The program requests the information if it is not given on the command line. If the file system is too large to fit on one reel, *volcopy* will prompt for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two drives.

The *fsname* argument represents the mounted name (e.g.: *root*, *u1*, etc.) of the filesystem being copied.

The *special* should be the physical disk section or tape (e.g.: */dev/rrp15*, */dev/rmt0*, */dev/rst0*, etc.).

The *volname* is the physical volume name (e.g.: *pk3*, *t0122*, etc.) and should match the external label sticker. Such label names are limited to five or fewer characters.

*Special1* and *volname1* are the device and volume from which the copy of the file system is being extracted. *Special2* and *volname2* are the target device and volume.

*Fsname* and *volname* are recorded in the last 12 characters of the superblock (char *fsname*[6], *volname*[6];).

*Labelit* can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, *labelit* prints current label values. The -n option provides for initial labeling of new tapes only (this destroys previous contents).

## FILES

*/etc/log/filesave* a record of file systems/volumes copied

## SEE ALSO

*filsys*(5).

## BUGS

Only device names beginning */dev/rmt* and */dev/rst* are treated as tapes.





**NAME**

**whatconf** — what device drivers are in an unix kernel

**SYNOPSIS**

**/etc/whatconf** unixkernel

**DESCRIPTION**

*Whatconf* tells you what devices the specified unix kernel is configured for.

**SEE ALSO**

**newconf(1m)**

100-100000

100-100000

100-100000

100-100000



## NAME

whodo - who is doing what

## SYNOPSIS

/etc/whodo

## DESCRIPTION

*Whodo* produces merged, reformatted, and dated output from the *who*(1) and *ps*(1) commands.

## SEE ALSO

*ps*(1), *who*(1).

(to be filled in)

Page 1

Date

1. The first part of the report is a summary of the work done during the last year. This includes a description of the work done, the results obtained, and a discussion of the work.

2. The second part of the report is a description of the work done during the last year. This includes a description of the work done, the results obtained, and a discussion of the work.



## NAME

xd - hexadecimal dump

## SYNOPSIS

xd [ file ] [ offset1[ . ][ b ] [ - [offset2[ . ][ b ] ]

## DESCRIPTION

*Xd* dumps *file* in hexadecimal and as characters.

The *file* argument specifies which file is to be dumped. If no file argument is specified, the standard input is used.

The *offset* argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as hexadecimal bytes. If it starts with '0', the offset is interpreted in octal. If '.' is appended, the offset is interpreted in decimal. If 'b' is appended, the offset is interpreted in blocks of 512 bytes. If the file argument is omitted, the offset argument must be preceded '+'.

If no second offset is specified, dumping continues until end-of-file. Otherwise dumping ends *before* the specified endaddress. The interpretation of *offset2* is the same as above.

## SEE ALSO

adb(1)





# NAME

*tgetent*, *tgetnum*, *tgetflag*, *tgetstr*, *tgoto*, *tputs* — terminal independent operation routines

# SYNOPSIS

```
char PC;
char *BC;
char *UP;
short ospeed;

tgetent(bp, name)
char *bp, *name;

tgetnum(id)
char *id;

tgetflag(id)
char *id;

char *
tgetstr(id, area)
char *id, **area;

char *
tgoto(cm, destcol, destline)
char *cm;

tputs(cp, affcnt, outc)
register char *cp;
int affcnt;
int (*outc)();
```

# DESCRIPTION

These functions extract and use capabilities from the terminal capability data base *termcap(5)*. These are low level routines; see *curses(3)* for a higher level package.

*Tgetent* extracts the entry for terminal *name* into the buffer at *bp*. *Bp* should be a character buffer of size 1024 and must be retained through all subsequent calls to *tgetnum*, *tgetflag*, and *tgetstr*. *Tgetent* returns -1 if it cannot open the *termcap* file, 0 if the terminal name given does not have an entry, and 1 if all goes well. It will look in the environment for a TERMCAP variable. If found, and the value does not begin with a slash, and the terminal type name is the same as the environment string TERM, the TERMCAP string is used instead of reading the *termcap* file. If it does begin with a slash, the string is used as a path name rather than */etc/termcap*. This can speed up entry into programs that call *tgetent*, as well as to help debug new terminal descriptions or to make one for your terminal if you can't write the file */etc/termcap*.

*Tgetnum* gets the numeric value of capability *id*, returning -1 if is not given for the terminal. *Tgetflag* returns 1 if the specified capability is present in the terminal's entry, 0 if it is not. *Tgetstr* gets the string value of capability *id*, placing it in the buffer at *area*, advancing the *area* pointer. It decodes the abbreviations for this field described in *termcap(5)*, except for cursor addressing and padding information.

*Tgoto* returns a cursor addressing string decoded from *cm* to go to column *destcol* in line *destline*. It uses the external variables UP (from the up capability) and BC (if bc is given rather than bs) if necessary to avoid placing \n, ^D or ^@ in the returned string. (Programs which call *tgoto* should be sure to turn off the XTABS bit(s), since *tgoto* may now output a tab. Note that programs using *termcap* should in general turn off XTABS anyway since some terminals use control I for other functions, such as nondestructive space.) If a % sequence is given which is not understood, then *tgoto* returns "OOPS".





*Tputs* decodes the leading padding information of the string *cp*; *affcnt* gives the number of lines affected by the operation, or 1 if this is not applicable, *putc* is a routine which is called with each character in turn. The external variable *ospeed* should contain the output speed of the terminal as encoded by *sny* (2). The external variable *PC* should contain a pad character to be used (from the *pc* capability) if a null ("@") is inappropriate.

**FILES**

/usr/lib/libtermcap.a -libtermcap library  
/etc/termcap data base

**SEE ALSO**

ex(1), curses(3), termcap(5)

**AUTHOR**

William Joy

**BUGS**

The following information was obtained from the records of the Department of the Interior, Bureau of Land Management, for the year ending December 31, 1964.

The total number of acres of land owned by the United States is 1,000,000,000. The total number of acres of land owned by the State of California is 100,000,000. The total number of acres of land owned by the State of Texas is 100,000,000.

The total number of acres of land owned by the State of New York is 100,000,000. The total number of acres of land owned by the State of Illinois is 100,000,000. The total number of acres of land owned by the State of Ohio is 100,000,000.

The total number of acres of land owned by the State of Michigan is 100,000,000. The total number of acres of land owned by the State of Indiana is 100,000,000. The total number of acres of land owned by the State of Kentucky is 100,000,000.

The total number of acres of land owned by the State of Tennessee is 100,000,000. The total number of acres of land owned by the State of Mississippi is 100,000,000. The total number of acres of land owned by the State of Alabama is 100,000,000.

The total number of acres of land owned by the State of Georgia is 100,000,000. The total number of acres of land owned by the State of Florida is 100,000,000. The total number of acres of land owned by the State of Louisiana is 100,000,000.

The total number of acres of land owned by the State of Arkansas is 100,000,000. The total number of acres of land owned by the State of Missouri is 100,000,000. The total number of acres of land owned by the State of Iowa is 100,000,000.

The total number of acres of land owned by the State of Nebraska is 100,000,000. The total number of acres of land owned by the State of Kansas is 100,000,000. The total number of acres of land owned by the State of Oklahoma is 100,000,000.



## NAME

hk - RK611/RK06, RK07 moving-head disk

## DESCRIPTION

The octal representation of the minor device number is encoded *dp*, where *d* is a physical drive number, and *p* is a logical unit (subsection) within a physical unit. The origins and sizes of the logical units on each drive, counted in cylinders of 66 512-byte blocks, are:

log. unit	start cyl.	length	size (in blocks)	
0	0	146	9636	(root on RK06/07)
1	146	135	8910	(swap on RK06/07)
2	281	129	8514	(rest of RK06)
3	281	533	35178	(rest of RK07)
4	0	0	0	(spare)
5	0	0	0	(spare)
6	0	410	27060	(all of RK06)
7	0	814	53724	(all of RK07)

Systems distributed for these devices use disk 0 for the root, disk 1 for swapping, and disk 6 (RK06) or disk 7 (RK07) for a mounted user file system.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records.

A 'raw' interface provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.' In raw I/O the buffer must begin on a word boundary.

## FILES

/dev/hk?, /dev/rhk?

## SEE ALSO

format (8)

## BUGS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*, *write* and *lseek*(2) should always deal in 512-byte multiples.

MEMORANDUM FOR THE RECORD

Subject: [Illegible text]

[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]

[Illegible text]

[Illegible text]

[Illegible text]

[Illegible text]

[Illegible text]

[Illegible text]

[Illegible text]

[Illegible text]



## NAME

hp - RP04/05/06, RM02/03 moving-head disk

## DESCRIPTION

The octal representation of the minor device number is encoded *dp*, where *d* is a physical drive number, and *p* is a logical unit (subsection) within a physical unit. The origins and sizes of the logical units on each drive, counted in cylinders of 418 512-byte blocks, for the RP04/05/06 are:

log. unit	start cyl.	length	size (in blocks)	
0	0	23	9614	(root on RP04/05/06)
1	23	21	8778	(swap on RP04/05/06)
2	44	21	8778	(/sys on RP04/05/06)
3	65	345	144210	(rest of RP04/05)
4	65	749	313082	(rest of RP06)
5	411	403	168454	(/usr on RP06)
6	0	410	171380	(all of RP04/05)
7	0	814	340252	(all of RP06)

The logical units for the RM02/03 are:

log. unit	start cyl.	length	size (in blocks)	
0	0	60	9600	(root on RM02/03)
1	60	55	8800	(swap on RM02/03)
2	115	50	8000	(/sys on RM02/03)
3	165	657	105120	(rest of RM02/03)
4	0	0	0	(spare)
5	0	0	0	(spare)
6	0	0	0	(spare)
7	0	822	131520	(all of RM02/03)

Systems distributed for these devices use disk 0 for the root, disk 1 for swapping, and disk 3 (RP04/05, RM02/03) or disk 4 (RP06) for a mounted user file system.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records.

A 'raw' interface provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.' In raw I/O the buffer must begin on a word boundary.

## FILES

/dev/rp?, /dev/rrp?  
/dev/rm?, /dev/rrm?

## SEE ALSO

format (8)

## BUGS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*, *write* and *lseek*(2)

... ..  
 ... ..  
 ... ..  
 ... ..

...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

... ..  
 ... ..  
 ... ..  
 ... ..  
 ... ..  
 ... ..  
 ... ..  
 ... ..  
 ... ..  
 ... ..

... ..  
 ... ..  
 ... ..

... ..  
 ... ..  
 ... ..

... ..  
 ... ..  
 ... ..



should always deal in 512-byte multiples.





## NAME

*rx* - RX01 or RX02 floppy disk

## DESCRIPTION

The *rx* driver supports both double sided (DS) or single sided (SS) drives and double density (DD) or single density (SD) format.

*Rx?* refers to an entire disk as a single sequentially-addressed file. Its 256-word blocks are numbered 0 to (number of blocks - 1). For the number of blocks on a floppy disk see the following table:

500	SS/SD
1000	DS/SD
1001	SS/DD
2002	DS/DD

*Rx0* refer to drive 0, single density format, *rx1* to drive 1, single density format, *rx2* to drive 0, double density format and *rx3* refer to drive 1, double density format.

The *rx* files discussed above access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a 'raw' interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw RX files begin with *rrx*.

In raw I/O the buffer must begin on a word boundary, and counts should be a multiple of 512 bytes (a disk block).

## FILES

/dev/*rx?*, /dev/*rrx?*

## SEE ALSO

*rxctrl* (1), *format* (8)

## BUGS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*, *write* and *lseek*(2) should always deal in 512-byte multiples.

THE NEW YORK PUBLIC LIBRARY  
ASTOR LENOX TILDEN FOUNDATION  
1054 FIFTH AVENUE, NEW YORK, N. Y. 10028

THE NEW YORK PUBLIC LIBRARY  
ASTOR LENOX TILDEN FOUNDATION  
1054 FIFTH AVENUE, NEW YORK, N. Y. 10028

THE NEW YORK PUBLIC LIBRARY  
ASTOR LENOX TILDEN FOUNDATION  
1054 FIFTH AVENUE, NEW YORK, N. Y. 10028



## NAME

st - SCT11 Streamer interface

## DESCRIPTION

The special files *rst0*, *nrst0* refer to the streamer drive 0. The letter *r* indicates "raw" device, the letter *n* indicates "no rewind" when the streamer is closed. For *nrst0* the bit 0200 in the minor device number must be set. To rewind the streamer tape you can say "`< /dev/rst0`".

The raw devices *rst0* and *nrst0* allow transfers of up to 127 512-byte blocks with a single *read* or *write* call. The byte count should always be an exact multiple of 512.

Unfortunately, when copying blocks from disk to tape, the streamer needs data faster than MUNIX can provide. Therefore, the streamer over- or underruns its internal buffers and does not stream. This means that after it has transferred some data, the tape will stop. When subsequent data arrives, it will rewind a piece of tape, then read forward to where it stopped, and immediately write the data. This zigzag-motion of the tape can be very time-consuming. The "zig" occurs when transferring data, and its time is proportional to the amount of data transferred. The "zag" is the rewind, and its time is constant. The ratio of "zig"-time to "zag"-time becomes tolerable, when the amount of data transferred with one "zig" gets large.

Two programs have been modified to use larger buffersizes when working with the streamer: *cpio* and *volcopy*. For both programs, the option `-S` sets the blocking factor to high values (120 for *cpio*, 110 for *volcopy*). The program *volcopy* now also exists in a standalone version. These programs allow you to backup your disks properly. Once in a while you should make a physical copy of your root file system with *volcopy*. The standalone version of *volcopy* can then be used to recreate a root file system after a catastrophic failure. At least once a day you should make a total or incremental dump of all files with *cpio*. From the *cpio*-tapes you can easily retrieve single files or whole directories.

## EXAMPLES

## total dump:

```
find / -print | cpio -oS >/dev/rst0
```

## incremental dump (last three days, say)

```
find / -mtime -3 -print | cpio -oS >/dev/rst0
```

## file retrieval:

```
cpio -ivS myfile </dev/rst0
```

## physical dump:

```
labelit /dev/rst0 root tape1 -n
```

```
volcopy -S root /dev/rhk0 hk0 /dev/rst0 tape1
```

## standalone (only terminal input shown):

```
/sa/volcopy
```

```
g0
```

```
-S root st(0,0) tape1 hk(0,0) root
```

## SEE ALSO

*volcopy*(1), *cpio*(1), *labelit*(1), *stctrl*(1)

## BUGS

CONFIDENTIAL

The following information was obtained from a confidential source who has provided reliable information in the past. It is being provided to you for your information only and should not be disseminated outside your office.

The source has advised that the following information was obtained from a confidential source who has provided reliable information in the past. It is being provided to you for your information only and should not be disseminated outside your office.

The source has advised that the following information was obtained from a confidential source who has provided reliable information in the past. It is being provided to you for your information only and should not be disseminated outside your office.

The source has advised that the following information was obtained from a confidential source who has provided reliable information in the past. It is being provided to you for your information only and should not be disseminated outside your office.

The source has advised that the following information was obtained from a confidential source who has provided reliable information in the past. It is being provided to you for your information only and should not be disseminated outside your office.



- 1) While rewinding to Begin Of Tape (BOT) the system pauses.
- 2) Using *nrst0* in subsequent calls from command scripts can cause trouble. A *sleep 2* after each streamer operation will help. For example: (making a cartridge with standalone programs)

```
< /dev/rst0
cat /sa/boot >/dev/nrst0
sleep 2
cat /sa/volcopy >/dev/nrst0
sleep 2
cat /sa/emuformat >/dev/rst0
```

The same is true for programmed I/O. After a *close* your program should sleep about 2 seconds before opening the streamer again.

- 3) Do not care too much for the message *streamer error: cannot read status*. In most cases the last streamer operation was completed successfully.
- 4)

---

(for your remarks)

As the Library is a public institution, it is the duty of the Trustees to see that the collection is maintained in a state of good order and that the books are accessible to the public.

REPORT

OF THE TRUSTEES

FOR THE YEAR 1888

The Trustees of the New York Public Library have the honor to acknowledge the receipt of the report of the Librarian for the year 1888, and to express their appreciation of the services rendered by him during the year.

The report of the Librarian shows that the collection has been maintained in a state of good order and that the books have been accessible to the public.



## NAME

*fstab* — static information about the filesystems

## SYNOPSIS

```
#include <fstab.h>
```

## DESCRIPTION

The file */etc/fstab* contains descriptive information about the various file systems. */etc/fstab* is only *read* by programs, and not written; it is the duty of the system administrator to properly create and maintain this file.

These programs use */etc/fstab*: *dump*, *mount*, *umount*, *swapon*, *fsck* and *df*. The order of records in */etc/fstab* is important, for both *fsck*, *mount*, and *umount* sequentially iterate through */etc/fstab* doing their thing.

The special file name is the block special file name, and not the character special file name. If a program needs the character special file name, the program must create it by appending a "r" after the last "/" in the special file name.

If *fs\_type* is "rw" or "ro" then the file system whose name is given in the *fs\_file* field is normally mounted read-write or read-only on the specified special file. The *fs\_freq* field is used for these file systems by the *dump(8)* command to determine which file systems need to be dumped. The *fs\_passno* field is used by the *fsck(8)* program to determine the order in which file system checks are done at reboot time. The root file system should be specified with a *fs\_passno* of 1, and other file systems should have larger numbers. File systems within a drive should have distinct numbers, but file systems on different drives can be checked on the same pass to utilize parallelism available in the hardware.

If *fs\_type* is "sw" then the special file is made available as a piece of swap space by the *swapon(8)* command at the end of the system reboot procedure. The fields other than *fs\_spec* and *fs\_type* are not used in this case.

*fs\_type* may be specified as "xx" to cause an entry to be ignored. This is useful to show disk partitions which are currently not used but will be used later.

```
#define FSTAB                "/etc/fstab"
#define FSNMLG               16

#define FSTABFMT             "%16s:%16s:%2s:%d:%d\n"
#define FSTABARG(p) (p) -> fs_spec, (p) -> fs_file, \
                      (p) -> fs_type, &(p) -> fs_freq, &(p) -> fs_passno
#define FSTABNARGS 5

#define FSTAB_RW             "rw"      /* read write device */
#define FSTAB_RO             "ro"      /* read only device */
#define FSTAB_SW             "sw"      /* swap device */
#define FSTAB_XX             "xx"      /* ignore totally */

struct fstab {
    char fs_spec[FSNMLG]; /* block special device name */
    char fs_file[FSNMLG]; /* file system path prefix */
    char fs_type[3];      /* rw,ro,sw or xx */
    int  fs_freq;         /* dump frequency, in days */
    int  fs_passno;       /* pass number on parallel dump */
};
```





The proper way to read records from */etc/fstab* is to use the routines `getfsent()`, `getfsspec()` or `getfsfile()`.

## FILES

*/etc/fstab*

## SEE ALSO

`getfsent(3)`





## NAME

termcap — terminal capability data base

## SYNOPSIS

/etc/termcap

## DESCRIPTION

*Termcap* is a data base describing terminals, used, e.g., by *w(1)* and *curses(3)*. Terminals are described in *termcap* by giving a set of capabilities which they have, and by describing how operations are performed. Padding requirements and initialization sequences are included in *termcap*.

Entries in *termcap* consist of a number of ':' separated fields. The first entry for each terminal gives the names which are known for the terminal, separated by '|' characters. The first name is always 2 characters long and is used by older version 6 systems which store the terminal type in a 16 bit word in a systemwide data base. The second name given is the most common abbreviation for the terminal, and the last name given should be a long name fully identifying the terminal. The second name should contain no blanks; the last name may well contain blanks for readability.

## CAPABILITIES

(P) indicates padding may be specified

(P=) indicates that padding may be based on no. lines affected

Name	Type	Pad?	Description
ae	str	(P)	End alternate character set
al	str	(P=)	Add new blank line
am	bool		Terminal has automatic margins
as	str	(P)	Start alternate character set
bc	str		Backspace if not ^H
bs	bool		Terminal can backspace with ^H
bt	str	(P)	Back tab
bw	bool		Backspace wraps from column 0 to last column
CC	str		Command character in prototype if terminal settable
cd	str	(P=)	Clear to end of display
ce	str	(P)	Clear to end of line
ch	str	(P)	Like cm but horizontal motion only, line stays same
cl	str	(P=)	Clear screen
cm	str	(P)	Cursor motion
co	num		Number of columns in a line
cr	str	(P=)	Carriage return, (default ^M)
cs	str	(P)	Change scrolling region (vt100), like cm
cv	str	(P)	Like ch but vertical only.
da	bool		Display may be retained above
dB	num		Number of millisec of bs delay needed
db	bool		Display may be retained below
dC	num		Number of millisec of cr delay needed
dc	str	(P=)	Delete character
dF	num		Number of millisec of ff delay needed
dl	str	(P=)	Delete line
dm	str		Delete mode (enter)
dN	num		Number of millisec of nl delay needed
do	str		Down one line
dT	num		Number of millisec of tab delay needed
ed	str		End delete mode





ei	str		End insert mode; give ":ei=:" if lc
eo	str		Can erase overstrikes with a blank
ff	str	(P+)	Hardcopy terminal page eject (default "L")
hc	bool		Hardcopy terminal
hd	str		Half-line down (forward 1/2 linefeed)
ho	str		Home cursor (if no cm)
hu	str		Half-line up (reverse 1/2 linefeed)
hz	str		Hazeltine; can't print ""s
ic	str	(P)	Insert character
if	str		Name of file containing is
im	bool		Insert mode (enter); give ":im=:" if lc
in	bool		Insert mode distinguishes nulls on display
ip	str	(P+)	Insert pad after character inserted
is	str		Terminal initialization string
k0-k9	str		Sent by "other" function keys 0-9
kb	str		Sent by backspace key
kd	str		Sent by terminal down arrow key
ke	str		Out of "keypad transmit" mode
kh	str		Sent by home key
kl	str		Sent by terminal left arrow key
kn	num		Number of "other" keys
ko	str		Termcap entries for other non-function keys
kr	str		Sent by terminal right arrow key
ks	str		Put terminal in "keypad transmit" mode
ku	str		Sent by terminal up arrow key
l0-l9	str		Labels on "other" function keys
li	num		Number of lines on screen or page
ll	str		Last line, first column (if no cm)
ma	str		Arrow key map, used by vi version 2 only
mi	bool		Safe to move while in insert mode
ml	str		Memory lock on above cursor.
mu	str		Memory unlock (turn off memory lock).
nc	bool		No correctly working carriage return (DM2500,H2000)
nd	str		Non-destructive space (cursor right)
nl	str	(P+)	Newline character (default \n)
ns	bool		Terminal is a CRT but doesn't scroll.
os	bool		Terminal overstrikes
pc	str		Pad character (rather than null)
pt	bool		Has hardware tabs (may need to be set with is)
se	str		End stand out mode
sf	str	(P)	Scroll forwards
sg	num		Number of blank chars left by so or se
so	str		Begin stand out mode
sr	str	(P)	Scroll reverse (backwards)
ta	str	(P)	Tab (other than "I or with padding)
tc	str		Entry of similar terminal - must be last
te	str		String to end programs that use cm
ti	str		String to begin programs that use cm
uc	str		Underscore one char and move past it
ue	str		End underscore mode
ug	num		Number of blank chars left by us or ue
ul	bool		Terminal underlines even though it doesn't overstrike





up	str	Upline (cursor up)
us	str	Start underscore mode
vb	str	Visible bell (may not move cursor)
ve	str	Sequence to end open/visual mode
vs	str	Sequence to start open/visual mode
xb	bool	Beehive (f1=escape, f2=ctrl C)
xn	bool	A newline is ignored after a wrap (Concept)
xr	bool	Return acts like ce \r \n (Delta Data)
xs	bool	Standout not erased by writing over it (HP 264?)
xt	bool	Tabs are destructive, magic so char (Telaray 1061)

### A Sample Entry

The following entry, which describes the Concept-100, is among the more complex entries in the *termcap* file as of this writing. (This particular concept entry is outdated, and is used as an example only.)

```
c1|c100|concept100:is=\EU\E\E7\E5\E8\EI\ENH\EK\E\200\Eo&\200:\
:al=3-\E\R:am:bs:cd=16-\E\C:ce=16\E\S:cl=2-\L:cm=\Ea%+ %+ :co#80:\
:dc=16\E^A:dl=3-\E^B:ei=\E\200:eo:im=\E^P:in:ip=16-\li#24:mi:nd=\E=\
:se=\Ed\Ee:so=\ED\EE:ta=8\t:ul:up=\E::vb=\Ek\EK:xn:
```

Entries may continue onto multiple lines by giving a \ as the last character of a line, and that empty fields may be included for readability (here between the last field on a line and the first field on the next). Capabilities in *termcap* are of three types: Boolean capabilities which indicate that the terminal has some particular feature, numeric capabilities giving the size of the terminal or the size of particular delays, and string capabilities, which give a sequence which can be used to perform particular terminal operations.

### Types of Capabilities

All capabilities have two letter codes. For instance, the fact that the Concept has "automatic margins" (i.e. an automatic return and linefeed when the end of a line is reached) is indicated by the capability *am*. Hence the description of the Concept includes *am*. Numeric capabilities are followed by the character '#' and then the value. Thus *co* which indicates the number of columns the terminal has gives the value '80' for the Concept.

Finally, string valued capabilities, such as *ce* (clear to end of line sequence) are given by the two character code, an '=', and then a string ending at the next following ':'. A delay in milliseconds may appear after the '=' in such a capability, and padding characters are supplied by the editor after the remainder of the string is sent to provide this delay. The delay can be either a integer, e.g. '20', or an integer followed by an '.', i.e. '3.'. A '.' indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. When a '.' is specified, it is sometimes useful to give a delay of the form '3.5' specify a delay per unit to tenths of milliseconds.

A number of escape sequences are provided in the string valued capabilities for easy encoding of characters there. A \E maps to an ESCAPE character, ^x maps to a control-x for any appropriate x, and the sequences \n \r \t \b \f give a newline, return, tab, backspace and formfeed. Finally, characters may be given as three octal digits after a \, and the characters ^ and \ may be given as \^ and \\. If it is necessary to place a : in a capability it must be escaped in octal as \072. If it is necessary to place a null character in a string capability it must be encoded as \200. The routines which deal with *termcap* use C strings, and strip the high bits of the output very late so that a \200 comes out as a \000 would.





### Preparing Descriptions

We now outline how to prepare descriptions of terminals. The most effective way to prepare a terminal description is by imitating the description of a similar terminal in *termcap* and to build up a description gradually, using partial descriptions with *ex* to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the *termcap* file to describe it or bugs in *ex*. To easily test a new terminal description you can set the environment variable *TERMCAP* to a pathname of a file containing the description you are working on and the editor will look there rather than in */etc/termcap*. *TERMCAP* can also be set to the *termcap* entry itself to avoid reading the file when starting up the editor. (This only works on version 7 systems.)

### Basic capabilities

The number of columns on each line for the terminal is given by the *co* numeric capability. If the terminal is a CRT, then the number of lines on the screen is given by the *li* capability. If the terminal wraps around to the beginning of the next line when it reaches the right margin, then it should have the *am* capability. If the terminal can clear its screen, then this is given by the *cl* string capability. If the terminal can backspace, then it should have the *bs* capability, unless a backspace is accomplished by a character other than *^H* (ugh) in which case you should give this character as the *be* string capability. If it overstrikes (rather than clearing a position when a character is struck over) then it should have the *os* capability.

A very important point here is that the local cursor motions encoded in *termcap* are undefined at the left and top edges of a CRT terminal. The editor will never attempt to backspace around the left edge, nor will it attempt to go up locally off the top. The editor assumes that feeding off the bottom of the screen will cause the screen to scroll up, and the *am* capability tells whether the cursor sticks at the right edge of the screen. If the terminal has switch selectable automatic margins, the *termcap* file usually assumes that this is on, i.e. *am*.

These capabilities suffice to describe hardcopy and "glass-tty" terminals. Thus the model 33 teletype is described as

```
t3|33|tty33:co#72:os
```

while the Lear Siegler ADM-3 is described as

```
d|adm3|lsi adm3:am:bs:cl="Z:li#24:co#80
```

### Cursor addressing

Cursor addressing in the terminal is described by a *cm* string capability, with *printf(3s)* like escapes *%x* in it. These substitute to encodings of the current line or column position, while other characters are passed through unchanged. If the *cm* string is thought of as being a function, then its arguments are the line and then the column to which motion is desired, and the *%* encodings have the following meanings:

<i>%d</i>	as in <i>printf</i> , 0 origin
<i>%2</i>	like <i>%2d</i>
<i>%3</i>	like <i>%3d</i>
<i>%.</i>	like <i>%c</i>
<i>%+x</i>	adds <i>x</i> to value, then <i>%</i> .
<i>%&gt;xy</i>	if value > <i>x</i> adds <i>y</i> , no output.
<i>%r</i>	reverses order of line and column, no output
<i>%i</i>	increments line/column (for 1 origin)
<i>%%</i>	gives a single <i>%</i>
<i>%n</i>	exclusive or row and column with 0140 (DM2500)
<i>%B</i>	BCD (16-( <i>x</i> /10)) + ( <i>x</i> %10), no output.
<i>%D</i>	Reverse coding ( <i>x</i> -2-( <i>x</i> %16)), no output. (Delta Data).

The first part of the report deals with the general situation of the country. It is a very interesting and informative study of the country's development. The author has done a great deal of research and has gathered a wealth of material. The report is well written and is a valuable contribution to the study of the country's development.

The second part of the report deals with the economic situation of the country. It is a very interesting and informative study of the country's economic development. The author has done a great deal of research and has gathered a wealth of material. The report is well written and is a valuable contribution to the study of the country's economic development.

The third part of the report deals with the social situation of the country. It is a very interesting and informative study of the country's social development. The author has done a great deal of research and has gathered a wealth of material. The report is well written and is a valuable contribution to the study of the country's social development.

The fourth part of the report deals with the political situation of the country. It is a very interesting and informative study of the country's political development. The author has done a great deal of research and has gathered a wealth of material. The report is well written and is a valuable contribution to the study of the country's political development.

1. General situation of the country	100
2. Economic situation of the country	100
3. Social situation of the country	100
4. Political situation of the country	100
5. Cultural situation of the country	100
6. Environmental situation of the country	100
7. Health situation of the country	100
8. Education situation of the country	100
9. Religion situation of the country	100
10. Other situation of the country	100

The fifth part of the report deals with the cultural situation of the country. It is a very interesting and informative study of the country's cultural development. The author has done a great deal of research and has gathered a wealth of material. The report is well written and is a valuable contribution to the study of the country's cultural development.



Consider the HP2645, which, to get to row 3 and column 12, needs to be sent `\Ea12c03Y` padded for 6 milliseconds. Note that the order of the rows and columns is inverted here, and that the row and column are printed as two digits. Thus its cm capability is `"cm=6\E&%r%2c%2Y"`. The Microterm ACT-IV needs the current row and column sent preceded by a "T", with the row and column simply encoded in binary, `"cm="T%.%."`. Terminals which use `"%."` need to be able to backspace the cursor (bs or bc), and to move the cursor up one line on the screen (up introduced below). This is necessary because it is not always safe to transmit `\t`, `\n` "D and `\r`, as the system may change or discard them.

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus `"cm=\E=%+ %+ "`.

### Cursor motions

If the terminal can move the cursor one position to the right, leaving the character at the current position unchanged, then this sequence should be given as `nd` (non-destructive space). If it can move the cursor up a line on the screen in the same column, this should be given as `up`. If the terminal has no cursor addressing capability, but can home the cursor (to very upper left corner of screen) then this can be given as `ho`; similarly a fast way of getting to the lower left hand corner can be given as `ll`; this may involve going up with `up` from the home position, but the editor will never do this itself (unless `ll` does) because it makes no assumption about the effect of moving up from the home position.

### Area clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as `ce`. If the terminal can clear from the current position to the end of the display, then this should be given as `ed`. The editor only uses `ed` from the first column of a line.

### Insert/delete line

If the terminal can open a new blank line before the line where the cursor is, this should be given as `al`; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as `dl`; this is done only from the first position on the line to be deleted. If the terminal can scroll the screen backwards, then this can be given as `sb`, but just `al` suffices. If the terminal can retain display memory above then the `da` capability should be given; if display memory can be retained below then `db` should be given. These let the editor understand that deleting a line on the screen may bring non-blank lines up from below or that scrolling back with `sb` may bring down non-blank lines.

### Insert/delete character

There are two basic kinds of intelligent terminals with respect to insert/delete character which can be described using *termcap*. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can find out which kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type `"abc def"` using local cursor motions (not spaces) between the `"abc"` and the `"def"`. Then position the cursor before the `"abc"` and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the `"abc"` shifts over to the `"def"` which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and should give the capability in, which stands for "insert null". If your terminal does something different and unusual then you



1. The first part of the report deals with the general situation of the country and the progress of the work during the year. It is divided into two main sections: the first section deals with the general situation of the country and the progress of the work during the year, and the second section deals with the specific work done during the year.

2. The second part of the report deals with the specific work done during the year. It is divided into three main sections: the first section deals with the work done in the field, the second section deals with the work done in the laboratory, and the third section deals with the work done in the office.

3. The third part of the report deals with the results of the work done during the year. It is divided into two main sections: the first section deals with the results of the field work, and the second section deals with the results of the laboratory work.

4. The fourth part of the report deals with the conclusions drawn from the work done during the year. It is divided into two main sections: the first section deals with the conclusions drawn from the field work, and the second section deals with the conclusions drawn from the laboratory work.

5. The fifth part of the report deals with the recommendations made by the committee. It is divided into two main sections: the first section deals with the recommendations made by the committee regarding the field work, and the second section deals with the recommendations made by the committee regarding the laboratory work.



may have to modify the editor to get it to use the insert mode your terminal defines. We have seen no terminals which have an insert mode not falling into one of these two classes.

The editor can handle both terminals which have an insert mode, and terminals which send a simple sequence to open a blank position on the current line. Give as `im` the sequence to get into insert mode, or give it an empty value if your terminal uses a sequence to insert a blank position. Give as `ei` the sequence to leave insert mode (give this, with an empty value also if you gave `im` so). Now give as `ic` any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give `ic`, terminals which send a sequence to open a screen position should give it here. (Insert mode is preferable to the sequence to open a position on the screen if your terminal has both.) If post insert padding is needed, give this as a number of milliseconds in `ip` (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in `ip`.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (e.g. if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability `mi` to speed up inserting in this case. Omitting `mi` will affect only speed. Some terminals (notably Datamedia's) must not have `mi` because of the way their insert mode works.

Finally, you can specify delete mode by giving `dm` and `ed` to enter and exit delete mode, and `dc` to delete a single character while in delete mode.

#### Highlighting, underlining, and visible bells

If your terminal has sequences to enter and exit standout mode these can be given as `so` and `se` respectively. If there are several flavors of standout mode (such as inverse video, blinking, or underlining — half bright is not usually an acceptable "standout" mode unless the terminal is in inverse video mode constantly) the preferred mode is inverse video by itself. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as the TVI 912 and Teleray 1061 do, this is acceptable, and although it may confuse some programs slightly, it can't be helped.

Codes to begin underlining and end underlining can be given as `us` and `ue` respectively. If the terminal has a code to underline the current character and move the cursor one space to the right, such as the Microterm Mime, this can be given as `uc`. (If the underline code does not move the cursor to the right, give the code followed by a nondestructive space.)

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement) then this can be given as `vb`; it must not move the cursor. If the terminal should be placed in a different mode during open and visual modes of `ex`, this can be given as `vs` and `ve`, sent at the start and end of these modes respectively. These can be used to change, e.g., from a underline to a block cursor and back.

If the terminal needs to be in a special mode when running a program that addresses the cursor, the codes to enter and exit this mode can be given as `ti` and `te`. This arises, for example, from terminals like the Concept with more than one page of memory. If the terminal has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the terminal for cursor addressing to work properly.

If your terminal correctly generates underlined characters (with no special codes needed) even though it does not overstrike, then you should give the capability `ul`. If overstrikes are erasable with a blank, then this should be indicated by giving `eo`.

#### Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local (this applies, for example, to the unshifted HP 2621 keys). If the keypad can be set to



The first section of the report deals with the general situation of the country and the progress of the work. It is followed by a detailed account of the work done during the year, and a summary of the results. The report is divided into two main parts, the first of which deals with the general situation of the country and the progress of the work, and the second of which deals with the detailed account of the work done during the year, and a summary of the results.

The second section of the report deals with the detailed account of the work done during the year, and a summary of the results. It is divided into two main parts, the first of which deals with the general situation of the country and the progress of the work, and the second of which deals with the detailed account of the work done during the year, and a summary of the results.

The third section of the report deals with the detailed account of the work done during the year, and a summary of the results. It is divided into two main parts, the first of which deals with the general situation of the country and the progress of the work, and the second of which deals with the detailed account of the work done during the year, and a summary of the results.

The fourth section of the report deals with the detailed account of the work done during the year, and a summary of the results. It is divided into two main parts, the first of which deals with the general situation of the country and the progress of the work, and the second of which deals with the detailed account of the work done during the year, and a summary of the results.



transmit or not transmit, give these codes as *ks* and *ke*. Otherwise the keypad is assumed to always transmit. The codes sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as *kl*, *kr*, *ku*, *kd*, and *kh* respectively. If there are function keys such as *f0*, *f1*, ..., *f9*, the codes they send can be given as *k0*, *k1*, ..., *k9*. If these keys have labels other than the default *f0* through *f9*, the labels can be given as *l0*, *l1*, ..., *l9*. If there are other keys that transmit the same code as the terminal expects for the corresponding function, such as clear screen, the *termcap* 2 letter codes can be given in the *ko* capability, for example, *":ko=cl,ll,sf,sb:"*, which says that the terminal has clear, home down, scroll down, and scroll up keys that transmit the same thing as the *cl*, *ll*, *sf*, and *sb* entries.

The *ma* entry is also used to indicate arrow keys on terminals which have single character arrow keys. It is obsolete but still in use in version 2 of *vi*, which must be run on some minicomputers due to memory limitations. This field is redundant with *kl*, *kr*, *ku*, *kd*, and *kh*. It consists of groups of two characters. In each group, the first character is what an arrow key sends, the second character is the corresponding *vi* command. These commands are *h* for *kl*, *j* for *kd*, *k* for *ku*, *l* for *kr*, and *H* for *kh*. For example, the mime would be *:ma="Kj"Zk"Xl:* indicating arrow keys left (*"H*), down (*"K*), up (*"Z*), and right (*"X*). (There is no home key on the mime.)

#### Miscellaneous

If the terminal requires other than a null (zero) character as a pad, then this can be given as *pc*.

If tabs on the terminal require padding, or if the terminal uses a character other than *"I* to tab, then this can be given as *ta*.

Hazeltine terminals, which don't allow *" "* characters to be printed should indicate *hz*. Datamedia terminals, which echo carriage-return linefeed for carriage return and then ignore a following linefeed should indicate *nc*. Early Concept terminals, which ignore a linefeed immediately after an *am* wrap, should indicate *xn*. If an erase-eol is required to get rid of stand-out (instead of merely writing on top of it), *xs* should be given. Teleray terminals, where tabs turn all characters moved over to blanks, should indicate *xt*. Other specific terminal problems may be corrected by adding more capabilities of the form *xx*.

Other capabilities include *is*, an initialization string for the terminal, and *if*, the name of a file containing long initialization strings. These strings are expected to properly clear and then set the tabs on the terminal, if the terminal has settable tabs. If both are given, *is* will be printed before *if*. This is useful where *if* is */usr/lib/tabset/sid* but *is* clears the tabs first.

#### Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability *tc* can be given with the name of the similar terminal. This capability must be *last* and the combined length of the two entries must not exceed 1024. Since *termlib* routines search the entry from left to right, and since the *tc* capability is replaced by the corresponding entry, the capabilities given at the left override the ones in the similar terminal. A capability can be cancelled with *xx@* where *xx* is the capability. For example, the entry

```
hn|2621nl:ks@:ke@:tc=2621:
```

defines a 2621nl that does not have the *ks* or *ke* capabilities, and hence does not turn on the function key labels when in visual mode. This is useful for different modes for a terminal, or for different user preferences.

#### FILES

*/etc/termcap* file containing terminal descriptions



The first part of the document discusses the importance of maintaining accurate records. It states that without proper documentation, it is difficult to track progress and identify areas for improvement. The text emphasizes the need for a systematic approach to data collection and analysis.

In the second section, the author describes the methodology used for the study. This includes details about the sample size, data sources, and the statistical techniques employed. The goal is to ensure that the findings are based on reliable and valid information.

The third part of the document presents the results of the study. It shows that there is a significant correlation between the variables being studied. The data suggests that the proposed model is effective in predicting the outcomes.

Following the results, the author discusses the implications of the findings. It is noted that the study has several limitations, but the overall conclusions are robust. The research provides valuable insights into the field and offers suggestions for future work.

The final section of the document is a conclusion. It summarizes the key points of the study and reiterates the importance of the findings. The author expresses confidence in the results and their potential impact on the field.

References are listed at the end of the document, providing sources for the information used in the study. These include academic journals, books, and other relevant publications. The list is formatted according to standard academic conventions.

Appendix A

Appendix A contains additional data and figures that support the main text. It includes a detailed table of results and several charts that illustrate the trends in the data.

Appendix B

Appendix B provides further details about the study, including a list of abbreviations and a glossary of terms. This section is intended to help readers understand the terminology used throughout the document.



**SEE ALSO**

`ex(1)`, `curses(3)`, `termcap(3)`, `tset(1)`, `vi(1)`, `ul(1)`, `more(1)`

**AUTHOR**

William Joy

Mark Horton added underlining and keypad support

**BUGS**

*Ex* allows only 256 characters for string capabilities, and the routines in *termcap(3)* do not check for overflow of this buffer. The total length of a single entry (excluding only escaped newlines) may not exceed 1024.

The *ma*, *vs*, and *ve* entries are specific to the *vi* program.

Not all programs support all entries. There are entries that are not supported by any program.

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000

100-100000



**NAME**

*ttytype* — data base of terminal types by port

**SYNOPSIS**

/etc/ttytype

**DESCRIPTION**

*Ttytype* is a database containing, for each tty port on the system, the kind of terminal that is attached to it. There is one line per port, containing the terminal kind (as a name listed in *termcap* (5)), a space, and the name of the tty, minus /dev/.

This information is read by *tset*(1) and by *login*(1) to initialize the TERM variable at login time.

**SEE ALSO**

*tset*(1), *login*(1)

**BUGS**

Some lines are merely known as "dialup" or "plugboard".





**NAME**

startrek - THE game based on the t.v. series.

**SYNOPSIS**

/usr/games/startrek

**DESCRIPTION**

You are the captain of the starship Enterprise and you have to destroy a random number of klingons (typically 15-25) in 30 stardates. (A measure of time in space, think of it as a day.) Full instructions are given if you reply 'y' to DO YOU WANT INSTRUCTIONS? A brief list of instructions is given if you ever type in an illegal command.

If you reply 'p' to PILOT TRAINING OR REAL MISSION? the computer asks you for a task number. This is used to start the random number generator so you can play in the same galaxy again if you want to.

Docking at a starbase refuels and rearms the Enterprise. If you stop for repairs you are delayed one stardate. Waiting for repairs in space might also cost you time.

**BUGS**

The calculator returns distances slightly too large for inter-quadrant travel.

**FILES**

/usr/games/startrek object code  
/usr/lib/startrek instructions

**AUTHOR**

Originally written in Basic by Mike Mayfield, Centreline Engineering and extended by David Ahl of Creative Computing.  
Translated into C and extended by M.J.Bayliss UKC April-October 1977.





**NAME**

configuration information - table of interrupt vector and device addresses

**SYNOPSIS**

**cat /usr/sys/confinfo**

**DESCRIPTION**

*Confinfo* is a table of the interrupt vector and device addresses used in MUNIX. It contains: (see next page)

1. The first part of the document is a letter from the President of the United States to the Congress, dated January 1, 1863. It is a very important document, as it contains the President's annual message to Congress. The letter is written in a formal, dignified style, and it is one of the most important documents in American history.

2. The second part of the document is a report from the Secretary of the Interior, dated January 1, 1863. It is a very important document, as it contains the Secretary's annual report to the President. The report is written in a formal, dignified style, and it is one of the most important documents in American history.

3. The third part of the document is a report from the Secretary of the Treasury, dated January 1, 1863. It is a very important document, as it contains the Secretary's annual report to the President. The report is written in a formal, dignified style, and it is one of the most important documents in American history.

4. The fourth part of the document is a report from the Secretary of the War, dated January 1, 1863. It is a very important document, as it contains the Secretary's annual report to the President. The report is written in a formal, dignified style, and it is one of the most important documents in American history.

5. The fifth part of the document is a report from the Secretary of the Navy, dated January 1, 1863. It is a very important document, as it contains the Secretary's annual report to the President. The report is written in a formal, dignified style, and it is one of the most important documents in American history.



## NAME

format - how to format disks

## SYNOPSIS

```
/bin/rxctrl -f
/sa/rxformat
/sa/rlformat
/sa/rmformat
/sa/emuformat
/sa/xyloformat
```

## DESCRIPTION

**Rxctrl** formats a 5 1/4" floppy emulating a single sided, double density 8" floppy with a ANDROMEDA WDC11 controller. For further details see **RXCTRL(1)** the floppy driver manipulation program.

All the other formatting programs are standalone programs. Enter the Minitor (Type sync, push INIT) and type i.e.:

```
.rl          (load from RL02)
./sa/emuformat (executable file)
.g0         (start the program)
```

**Rxformat** formats a 8" floppy RX02 compatible. The floppy controller responses '\$'. Type...

```
XD2 (double density) or
XD1 (single density) <cr>
and
XU0 (left drive) or
XU1 (right drive) <cr>
```

**Rlformat** formats a whole TANDON TM603SE drive with a ANDROMEDA WDC11 controller as a RL02. The following arguments are interactively asked for:

```
UNIT:      0 (TANDON drive 0)
            1 (TANDON drive 1)
INTERLEAVE: 1..31 (odd)
            Use 5 to optimize the seek time.
```

**Rmformat** formats one or all tracks of a FUJITSU M2312K drive with a DATARAM S04/A controller as a RM02. The following arguments are interactively asked for:

```
SINGLE TRACK ? 'y' or <cr> if yes...
HEAD:         0..4
TRACK:        0..822
```

**Emuformat** formats a whole FUJITSU M2312K drive with a EMULEX SC02 controller as

```
Unit 0: RK07   Unit 3: RK07
Unit 1: RK07   Unit 4: RK07
Unit 2: RK06   Unit 5: RK06
```

1914

1915

1916

1917

1918

1919

1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930

1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030



## NAME

**fsck** — file system consistency check and interactive repair

## SYNOPSIS

```
/etc/fsck -p [ filesystem ... ]
/etc/fsck [ -y ] [ -n ] [ -sX ] [ -SX ] [ -t filename ] [ filesystem ] ...
```

## DESCRIPTION

The first form of *fsck* preens a standard set of filesystems or the specified file systems. It is normally used in the script */etc/re* during automatic reboot. In this case *fsck* reads the table */etc/fstab* to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible. Normally, the root file system will be checked on pass 1, other "root" ("a" partition) file systems on pass 2, other small file systems on separate passes (e.g. the "d" file systems on pass 3 and the "e" file systems on pass 4), and finally the large user file systems on the last pass, e.g. pass 5. A pass number of 0 in *fstab* causes a disk to not be checked; similarly partitions which are not shown as to be mounted "rw" or "ro" are not checked.

The system takes care that only a restricted class of innocuous inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

- Unreferenced inodes
- Link counts in inodes too large
- Missing blocks in the free list
- Blocks in the free list also in files
- Counts in the super-block wrong

These are the only inconsistencies which *fsck* with the *-p* option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature of the correction. After successfully correcting a file system, *fsck* will print the number of files on that file system and the number of used and free blocks.

Without the *-p* option, *fsck* audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that a number of the corrective actions which are not fixable under the *-p* option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond yes or no. If the operator does not have write permission *fsck* will default to a *-n* action.

*Fsck* has more consistency checks than its predecessors *check*, *dcheck*, *fcheck*, and *icheck* combined.

The following flags are interpreted by *fsck*.

- y* Assume a yes response to all questions asked by *fsck*; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.
- n* Assume a no response to all questions asked by *fsck*; do not open the file system for writing.
- sX* Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-



THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

TO THE HONORABLE CHAIRMAN OF THE BOARD OF TRUSTEES  
OF THE UNIVERSITY OF CHICAGO  
FROM  
THE DEPARTMENT OF CHEMISTRY  
SUBJECT: REPORT ON THE PROGRESS OF THE WORK OF THE DEPARTMENT OF CHEMISTRY DURING THE YEAR 1900-1901

The Department of Chemistry has during the year 1900-1901 been engaged in a large amount of research work, and has made considerable progress in many of its branches. The following is a summary of the work done during the year:

1. *Organic Chemistry* - The work of the Department of Organic Chemistry has been largely devoted to the study of the properties and reactions of the various classes of organic compounds. The following are some of the results of the work:

(a) *Aliphatic Compounds* - The work of the Department of Aliphatic Chemistry has been largely devoted to the study of the properties and reactions of the various classes of aliphatic compounds. The following are some of the results of the work:

(b) *Aromatic Compounds* - The work of the Department of Aromatic Chemistry has been largely devoted to the study of the properties and reactions of the various classes of aromatic compounds. The following are some of the results of the work:

(c) *Inorganic Chemistry* - The work of the Department of Inorganic Chemistry has been largely devoted to the study of the properties and reactions of the various classes of inorganic compounds. The following are some of the results of the work:

(d) *Physical Chemistry* - The work of the Department of Physical Chemistry has been largely devoted to the study of the properties and reactions of the various classes of physical compounds. The following are some of the results of the work:



core copy of the superblock will not continue to be used, or written on the file system.

The `-sX` option allows for creating an optimal free-list organization. The following forms of `X` are supported for the following devices:

- `-s3` (RP03)
- `-s4` (RP04, RP05, RP06)
- `-sBlocks-per-cylinder:Blocks-to-skip` (for anything else)

If `X` is not given, the values used when the filesystem was created are used. If these values were not specified, then the value `400:9` is used.

- `-SX` Conditionally reconstruct the free list. This option is like `-sX` above except that the free list is rebuilt only if there were no discrepancies discovered in the file system. Using `-S` will force a no response to all questions asked by `fsck`. This option is useful for forcing free list reorganization on uncontaminated file systems.
- `-t` If `fsck` cannot obtain enough memory to keep its tables, it uses a scratch file. If the `-t` option is specified, the file named in the next argument is used as the scratch file, if needed. Without the `-t` flag, `fsck` will prompt the operator for the name of the scratch file. The file chosen should not be on the filesystem being checked, and if it is not a special file or did not already exist, it is removed when `fsck` completes.

If no filesystems are given to `fsck` then a default list of file systems is read from the file `/etc/fstab`.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
  - Directory size not 16-byte aligned.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
  - File pointing to unallocated inode.
  - Inode number out of range.
8. Super Block checks:
  - More than 65536 inodes.
  - More blocks for inodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. The only restriction is that the directory `lost+found` must preexist in the root of the filesystem being checked and must have empty slots in which entries can be made. This is accomplished by making `lost+found`, copying a number of files to the directory, and then removing them (before `fsck` is executed).

Checking the raw device is almost always faster.

## FILES

`/etc/fstab` contains default list of file systems to check.

## DIAGNOSTICS

The diagnostics produced by `fsck` are intended to be self-explanatory.

THE UNIVERSITY OF CHICAGO  
DEPARTMENT OF THE HISTORY OF ARTS  
CHICAGO, ILLINOIS 60637

TO THE HONORABLE CHAIRMAN OF THE BOARD OF TRUSTEES  
OF THE UNIVERSITY OF CHICAGO  
FROM THE DEPARTMENT OF THE HISTORY OF ARTS

RE: REPORT OF THE DEPARTMENT OF THE HISTORY OF ARTS  
FOR THE YEAR 1967-1968

The Department of the History of Arts has the honor to acknowledge the receipt of your letter of the 10th day of May, 1968, and to inform you that the enclosed report contains the results of the department's activities during the year 1967-1968.

The report is divided into two parts: a summary of the department's activities and a list of the department's publications.

The summary of the department's activities is divided into three sections: a summary of the department's activities in the field of art history, a summary of the department's activities in the field of architectural history, and a summary of the department's activities in the field of the history of design.

The list of the department's publications is divided into two sections: a list of the department's publications in the field of art history, and a list of the department's publications in the field of architectural history.

The department's activities in the field of art history have been carried out in accordance with the plan of work for the year 1967-1968, which was approved by the Board of Trustees on the 10th day of May, 1967.

The department's activities in the field of architectural history have been carried out in accordance with the plan of work for the year 1967-1968, which was approved by the Board of Trustees on the 10th day of May, 1967.

The department's activities in the field of the history of design have been carried out in accordance with the plan of work for the year 1967-1968, which was approved by the Board of Trustees on the 10th day of May, 1967.

The department's activities in the field of art history, architectural history, and the history of design have been carried out in accordance with the plan of work for the year 1967-1968, which was approved by the Board of Trustees on the 10th day of May, 1967.

The department's activities in the field of art history, architectural history, and the history of design have been carried out in accordance with the plan of work for the year 1967-1968, which was approved by the Board of Trustees on the 10th day of May, 1967.

The department's activities in the field of art history, architectural history, and the history of design have been carried out in accordance with the plan of work for the year 1967-1968, which was approved by the Board of Trustees on the 10th day of May, 1967.



## NAME

getty - set typewriter mode

## SYNOPSIS

/etc/getty [ char ]

## DESCRIPTION

*Getty* is invoked by *init*(8) immediately after a typewriter is opened following a dial-up. It reads the user's login name and calls *login*(1) with the name as argument. While reading the name *getty* attempts to adapt the system to the speed and type of terminal being used.

*init* calls *getty* with a single character argument taken from the *ttys*(5) file entry for the terminal line, which is now ignored. *Getty* looks for the terminal line name with the *ttyname* call and then searches through the *ttysize* data base. This data base has 6 entries for each terminal line of the following form.

TERM	device	in speed	out speed	login mode	shell mode
pt80	tty0	300	300	PA;RW;CR1	PA;EC;TX;CR1
pv	tty10	9600	9600	PA;RW	PA;EC

The first entry is the *TERM* variable for the terminal type (see also *termcap* (5)), the second entry is the device name of the terminal line without '/dev/', the third and fourth entries are the terminal receive and transmit speed, the fifth and sixth are the tty modes for the login command and the normal shell mode. The entries must be separated by a ' ' or tab character, the tty modes by a ';' character. The meaning of the modes is as follows (see also *tty* (4)):

B0	BS0	DA	ALLDELAY
B1	BS1	DC	CRDELAY
CB	CBREAK	DB	BSDELAY
CM	CRMOD	DN	NLDELAY
CR0	CR0	DT	TBDELAY
CR1	CR1	DV	VTDELAY
CR2	CR2	N0	NL0
CR3	CR3	N1	NL1
PA	ANYP	N2	NL2
PE	EVENP	N3	NL3
PO	ODDP	EC	ECHO
T0	TAB0	TD	TANDEM
T1	TAB1	TX	XTABS
T2	TAB2	LC	LCASE
F0	FF0	F1	FF1
RW	RAW	HU	HPCL

When *getty* writes the '<node name> login:' greeting message to your terminal, answer with your user login name.

The user's name is terminated by a new-line or carriage-return character. In the second case CRMOD mode is set (see *ioctl*(2)).

The name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is nonempty, the system is told to map

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..



any future upper-case characters into the corresponding lower-case characters (Then \A is mapped to upper-case A).

Login is called with the user's name as argument.

**FILES**

/etc/ttys  
/etc/ttytype  
/etc/termcap

**SEE ALSO**

init(8), login(1), ioctl(2), ttys(5), termcap (5), tty (4)





**SEE ALSO**

*fstab*(5), *fs*(5), *crash*(8), *reboot*(8)

**BUGS**

Inode numbers for *.* and *..* in each directory should be checked for validity.

*-g* and *-b* options from *check* should be available in *fsck*.

There should be some way to start a *fsck -p* at pass *n*.

